ENCICLOPEDIA PRACTICA DE LA

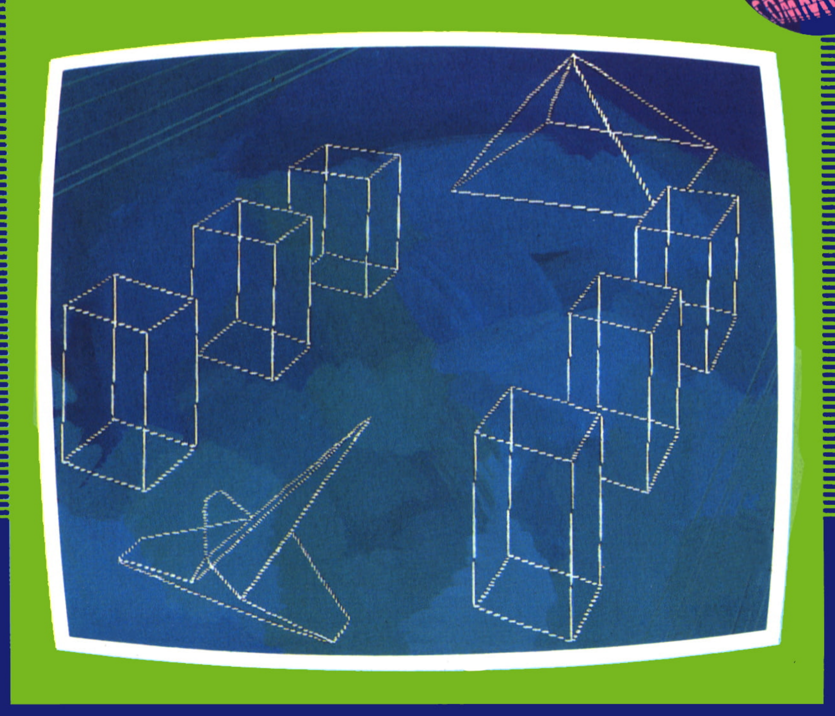
INFORMATICA APLICADA

9

Cómo hacer dibujos tridimensionales en el ordenador

Aula de Informática

SINCLAIR
AMSTRAD
COMMODORE
IBM



EDICIONES SIGLO CULTURAL

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

9

Cómo hacer dibujos tridimensionales en el ordenador

EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación
y Licenciado en Informática
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño y maquetación:
BRAVO-LOFISH.

Dibujos:
JOSE OCHOA Y ANTONIO PERERA.

Tomo 9. **Cómo hacer dibujos tridimensionales en el ordenador**
AULA DE INFORMATICA APLICADA (AIA)
FERNANDO SUERO, Diplomado de Telecomunicación
JOAQUIN SALVACHUA, Diplomado de Telecomunicación

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28020 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América. 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-033-2
ISBN de la obra: 84-7688-018-9.

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-39.894-1986
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid

Octubre, 1986.
P.V.P. Canarias: 365,-

INDICE

Los programas que aparecen en este libro funcionan en los ordenadores:

IBM–PC, XT, AT y compatibles.
AMSTRAD–464, 664, 6128, 1512.
SINCLAIR–SPECTRUM 48 K, 128 K, PLUS, PLUS 2.
MSX–Todos los modelos.
COMMODORE–CBM 64 y CBM 128.

INTRODUCCION: COMANDOS GRAFICOS 1



LA PANTALLA



AMOS a ver cómo está constituida la pantalla del ordenador y cómo podemos pintar sobre ella.

La pantalla está configurada como una retícula en la que cada casilla, o «pixel», puede tomar un color.

Para acceder a cada una de estas casillas necesitamos conocer su posición dentro de la pantalla, lo que indicaremos mediante la fila y la columna en la que está situada. A estos dos números nos referiremos como sus coordenadas.

El número de filas y columnas que hay, o resolución, varía de un ordenador a otro, por lo que los veremos por separado.

En el Spectrum la pantalla está formada por 176 filas y 256 columnas. El punto origen, que es el situado en la fila 0 y la columna 0, está en la esquina inferior izquierda de la pantalla.

En el Amstrad existen tres modos de pantalla a los que les corresponde diversos números de puntos. En el modo 0 existen 160 columnas y 200 filas. En el modo 1 hay 320 columnas y 200 filas. Finalmente, el modo 2 cuenta con 640 columnas y 200 filas.

Para unificar con los distintos modos el par de números que indican la fila y la columna, y que sean los mismos se varía el tamaño de la casilla, que ocupará varios puntos, por lo que a coordenadas distintas les puede corresponder una misma casilla.

En el modo 0 a una casilla le corresponderán 8 puntos de nuestras coordenadas (4*2), en el modo 1 cada casilla estará compuesta por 4 puntos (2*2), y en el modo 2 serán dos puntos (1*2).

Para especificar los diferentes modos existe el comando

MODE n

donde n es el modo a usar.

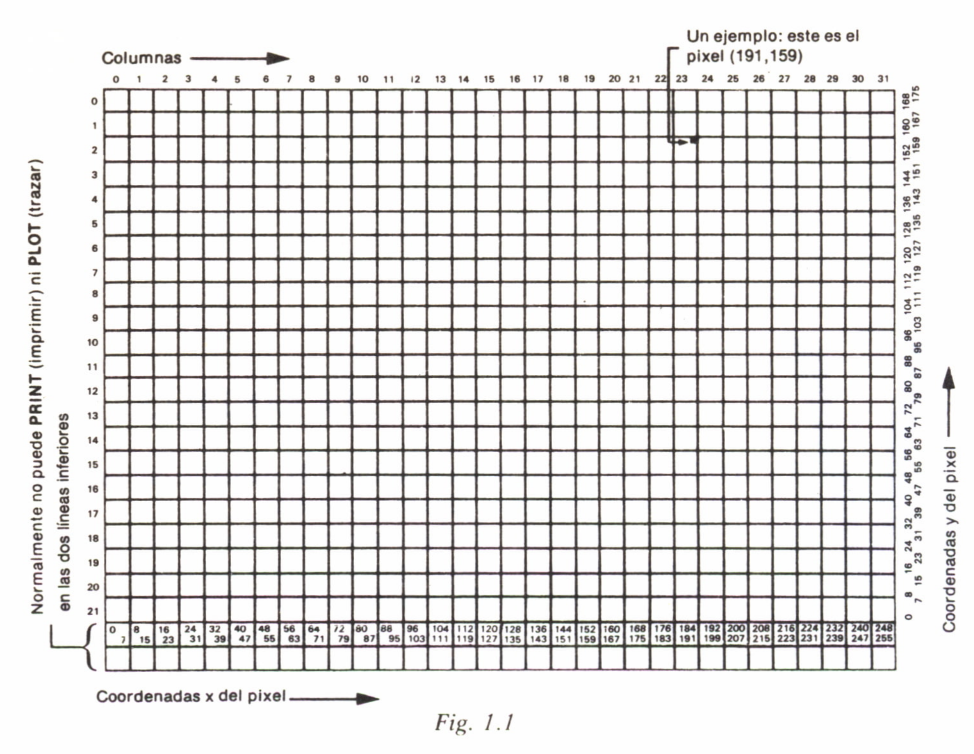


Fig. 1.1

Modo 0 — **Punto 4 × 2**

Modo 1 — **Punto 2 × 2**

Modo 2 — **Punto 1 × 2**

Fig. 1.2

El origen también está situado en la esquina inferior izquierda de la pantalla.

El Commodore tiene 320 columnas y 200 filas. Aquí el origen está situado en la esquina superior izquierda.

Este ordenador presenta un problema al no disponer de comandos gráficos, pero existe en el mercado una extensión que dispone de ellos (Basic Simon).

En el IBM también existen diversos formatos de pantalla. Un primer modo en el que hay 320 columnas y 200 filas, y un segundo en el que disponemos de 640 columnas y 200 filas. Estos modos se seleccionan con el comando

SCREEN n

donde n es el modo a seleccionar.

El origen está situado en la esquina superior izquierda.



COLORES

En la pantalla se puede pintar con diversos colores. Se puede especificar el color del borde, o marco, que es lo que rodea la pantalla y donde no se puede escribir; el color del fondo, o puntos sin pintar, y el de los puntos a escribir, o tinta.

Veamos cómo se indica esto en cada ordenador.

En el Spectrum existen 8 colores. A cada color le hacemos corresponder un código que lo identifique.

Estos serán:

Código	**Color**
0	Negro
1	Azul
2	Rojo
3	Magenta
4	Verde
5	Cyan
6	Amarillo
7	Blanco

Fig. 1.3. Códigos de colores para Spectrum.

Para indicar el color del borde indicamos:

BORDER n

donde n es el código correspondiente al color que queremos usar.

La instrucción:

PAPER n

cambiará el color del fondo.

Por último:

INK n

indicará el color de tinta con el que queremos pintar.

Hay que tener cuidado de no indicar el mismo color para el fondo y la tinta, pues de esta forma al pintar no será posible distinguirlo del fondo.

En el Amstrad existen 27 colores diferentes, aunque no se pueden usar todos simultáneamente.

El número de colores depende del modo que estemos usando, existiendo una relación inversa entre la resolución y el número de colores.

En el modo 0 se pueden usar 16 colores distintos. En el modo 1, 4 colores y, por último, en el modo 2 sólo 2 colores.

Código	**Color**	**Código**	**Color**
0	Negro	14	Azul pastel
1	Azul	15	Naranja
2	Azul intenso	16	Rosa
3	Rojo	17	Magenta pastel
4	Magenta	18	Verde intenso
5	Malva	19	Verde marino
6	Rojo intenso	20	Cian intenso
7	Morado	21	Verde lima
8	Magenta intenso	22	Verde pastel
9	Verde	23	Cian pastel
10	Cian	24	Dorado
11	Azul celeste	25	Amarillo pastel
12	Amarillo	26	Blanco intenso
13	Blanco		

Fig. 1.4 Códigos de colores para Amstrad.

El color del borde se asigna:

BORDER n

donde n es el código del color.

Para asignar el color del fondo y de lo que pintaremos usaremos las instrucciones

PAPER n PEN n

Pero aquí n no es el código del color, sino que se refiere al número del tintero. Esto del tintero es una forma indirecta de acceder al color que queremos usar.

Asignaremos el color del tintero mediante la instrucción

INK n,m

donde n es el número del tintero al que queremos asignar el valor y m el código del color que queremos asignar.

En el Commodore hay 16 colores.

Para el color del fondo debemos apretar la tecla «Ctrl» y una tecla del 1 al 8. Y para la tinta la tecla «Commodore» y una tecla del 1 al 8.

Teclas	Color	Teclas	Color
CTRL+1	Negro	C+1	Naranja
CTRL+2	Blanco	C+2	Marrón
CTRL+3	Rojo	C+3	Rojo intenso
CTRL+4	Cyan	C+4	Verde
CTRL+5	Púrpura	C+5	Verde
CTRL+6	Verde	C+6	Verde intenso
CTRL+7	Azul	C+7	Azul intenso
CTRL+8	Amarillo	C+8	Verde

Fig. 1.5 Códigos de colores para el Commodore.

En el IBM también depende el color del modo en el que estemos.

En el modo 1 podemos elegir 16 colores del fondo, que será el mismo del borde, y elegir entre 3 colores en dos paletas. Si seleccionamos la paleta 0 tendremos tres colores (verde, rojo y marrón), y de seleccionar la paleta 1 otros tres (cian, magenta y blanco).

Códigos	Colores	Códigos	Colores
0	Negro	9	Azul claro
1	Azul	10	Verde claro
2	Verde	11	Cian claro
3	Cian	12	Rojo claro
4	Rojo	13	Magenta claro
5	Magenta	14	Amarillo
6	Marrón	15	Blanco intenso
7	Blanco		
8	Gris		

Color	Paleta 0	Paleta 1
1	Verde	Cian
2	Rojo	Magenta
3	Marrón	Blanco

Fig. 1.6 Códigos para IBM.

En el modo 2 sólo hay 2 colores para pintar: 0 para el color del fondo y 1 para el blanco.



LINEAS Y PUNTOS

Una vez que sabemos usar los colores, veamos cómo pintar un punto.

Para ello usaremos la instrucción

PLOT X,Y,T

donde X e Y son las coordenadas del punto y T la tinta a usar. Esta instrucción es utilizable en el Amstrad y Commodore. Para utilizarla en el Spectrum debemos omitir el último parámetro (T).

En el Amstrad existe la instrucción

MOVE X,Y

que se sitúa en el punto indicado por X e Y sin pintar.

En el IBM usaremos

PSET (X,Y)

para pintar el punto situado en X e Y.

También existe la inversa

PRESET (X,Y)

que borrará el punto situado en X e Y pintándolo con el mismo color que el fondo.

Si queremos trazar una recta en el Spectrum utilizaremos la instrucción

DRAW X,Y

que unirá el último punto pintado con otro alejado de él X puntos en horizontal e Y puntos en vertical.

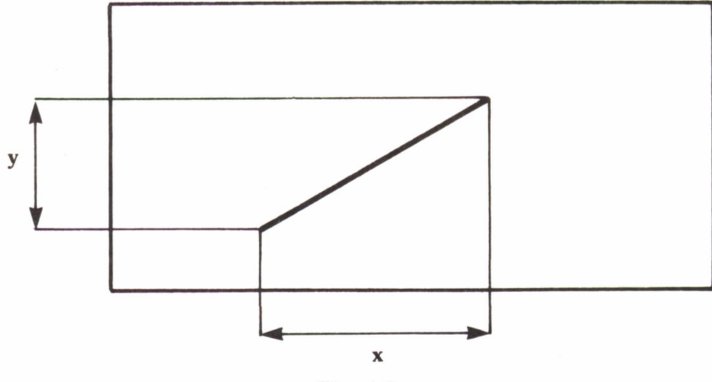


Fig. 1.7

La instrucción equivalente del AMSTRAD es

DRAWR X,Y

Además, existe la instrucción

DRAW X,Y

en la que la coordenada X e Y son referidas al punto 0,0 en vez de referirse al último punto pintado.

En el Commodore existe

LINE X1,Y1,X2,Y2,T

donde X1 e Y1 son las coordenadas del punto inicial y X2 e Y2 las del punto final indicando T si escribe (1) o borra (0).

Análoga a ésta es la instrucción del IBM

LINE (X1,Y1) - (X2,Y2)

que hace lo mismo que la del Commodore.

Por último, indicaremos que los programas no controlan que se intente dibujar fuera de la pantalla. En algunos ordenadores, como el Amstrad y el IBM, no ocurre nada, pero en otros, como el Spectrum y el Commodore, esto ocasionará un error. Si el lector desea evitarlo, antes de dibujar deberá controlar la posición de los puntos, y de salirse de la pantalla y dibujar sólo lo que entre en la pantalla, si lo hay. Este tipo de rutinas son conocidas como rutinas de «clipping».

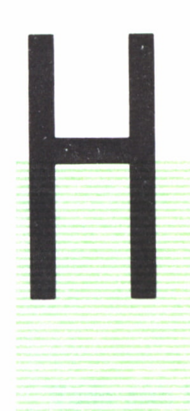
Esta rutina comprueba si el punto está fuera de la pantalla, y de ocurrir busca qué punto de la recta limita con el borde de la pantalla.

```
 10 REM ****************************
 20 REM **                        **
 30 REM **PROGRAMA PARA SPECTRUM **
 40 REM **                        **
 60 REM **    RUTINA DE CLIPPING **
 70 REM **                        **
 80 REM ****************************
 90 LET xi=0: LET yi=0
100 LET xf=255: LET yf=175
110 DIM x(2): DIM y(2)
120 REM LOS DATOS SON X1,Y1 Y X2,Y2
130 LET X(1)=X1: LET Y(1)=Y1
140 LET X(2)=X2: LET Y(2)=Y2
150 LET T=1: GO SUB 500
160 LET T=2: GO SUB 500
170 PLOT X(1),X(2)
```

```
180 DRAW X(2)-X(1),Y(2)-Y(1)
190 STOP
500 REM RUTINA PRINCIPAL
510 LET A=(Y(2)-Y(1))/(X(2)-X(1))
520 IF X(T)<XI THEN  LET Y(T)=Y(1)+(XI-X(1))*A
530 IF X(T)<XI THEN  LET X(T)=XI: GO TO 500
535 IF X(T)>XF THEN  LET Y(T)=Y(1)+(XF-X(1))*A
540 IF X(T)>XF THEN  LET X(T)=XF: GO TO 500
550 IF Y(T)>YF THEN  LET X(T)=X(1)+(YF-(1))/A
560 IF Y(T)>YF THEN  LET Y(T)=YF: GO TO 500
570 IF Y(T)<YI THEN  LET X(T)=X(1)+(YI-(1))/A
580 IF Y(T)<YI THEN  LET Y(T)=YI: GO TO 500
600 RETURN
```

Programa 1.1

EL MUNDO EN TRES DIMENSIONES 2



HASTA ahora hemos visto cómo dibujar en la pantalla de nuestro ordenador cualquier tipo de figuras, pero no hemos dado ningún tipo de significado a los puntos que vemos en la pantalla.

Tal y como está diseñado el ordenador, a cada punto de la pantalla podríamos hacer corresponder un punto del plano y así podríamos representar figuras de dos dimensiones en la pantalla. Para ello podemos hacer lo siguiente: tomamos un punto del plano que vamos a llamar «origen» y para poder colocar los puntos en el plano trazamos una flecha que vaya del origen a ese punto. O sea, lo que tenemos es un flecha que nos indica qué punto del plano vamos a pintar. A esta flecha se la denomina «vector».

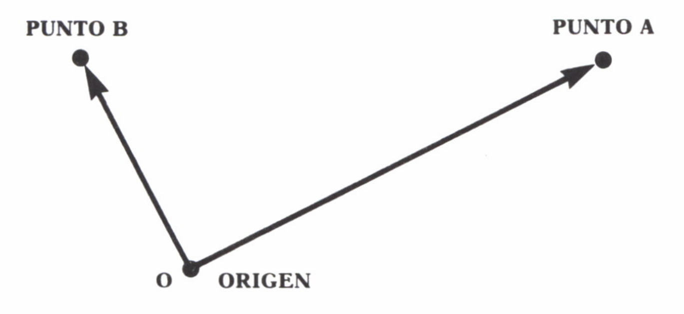


Fig. 2.1

Pero nuestro ordenador no entiende de «vectores» y «flechas», sino sólo de números y por ello le tendremos que traducir nuestras ideas a su lenguaje.

Esto lo vamos a hacer asignando a cada punto un par de números. El primero representa la distancia en horizontal, o eje X, a la que está el punto, con respecto al punto que hemos llamado origen, y el segundo la distancia en sentido vertical, o eje Y, a la que está el punto que queremos representar desde el origen.

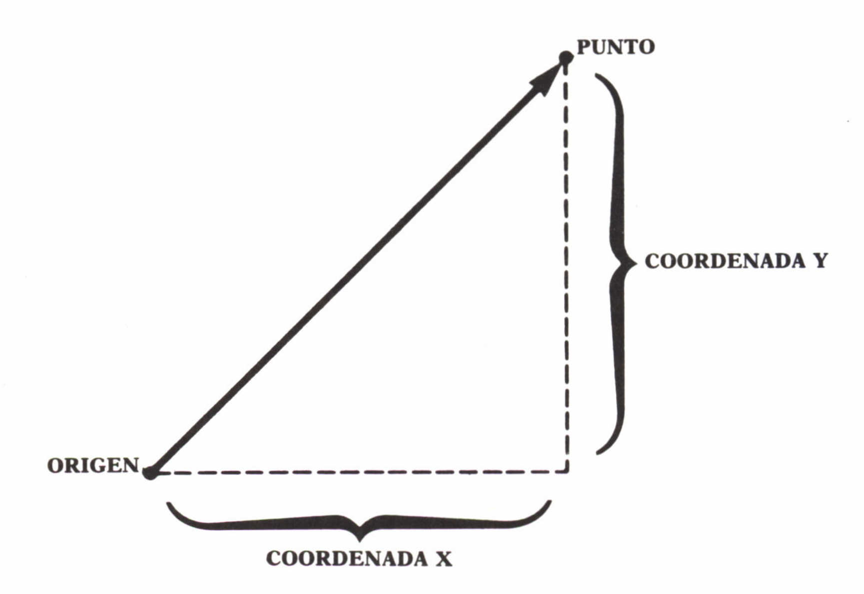


Fig. 2.2

Pues bien, a estos dos números se les llama coordenadas del vector. Evidentemente, el origen tiene como coordenadas el par (0,0), y los puntos que estén a la derecha del origen tendrán una coordenada horizontal, la primera del par, con un valor positivo, y los que estén a la izquierda serán negativos. Igualmente, todos aquellos puntos que estén por encima del origen tendrán una coordenada Y, en este caso el segundo elemento del par, positiva, y los que estén por debajo tendrán una coordenada Y negativa.

Este convenio de signos que hemos tomado nos divide el plano en cuatro partes.

La recta horizontal que parte del origen es donde vamos a tomar el valor de la coordenada X, por lo que se llama eje X, y la recta vertical que pasa por el origen donde tomamos el valor de la coordenada Y se llama eje Y.

Gracias a todo esto que hemos contado tenemos una forma de especificar cualquier punto del plano mediante un par de números, sus coordenadas.

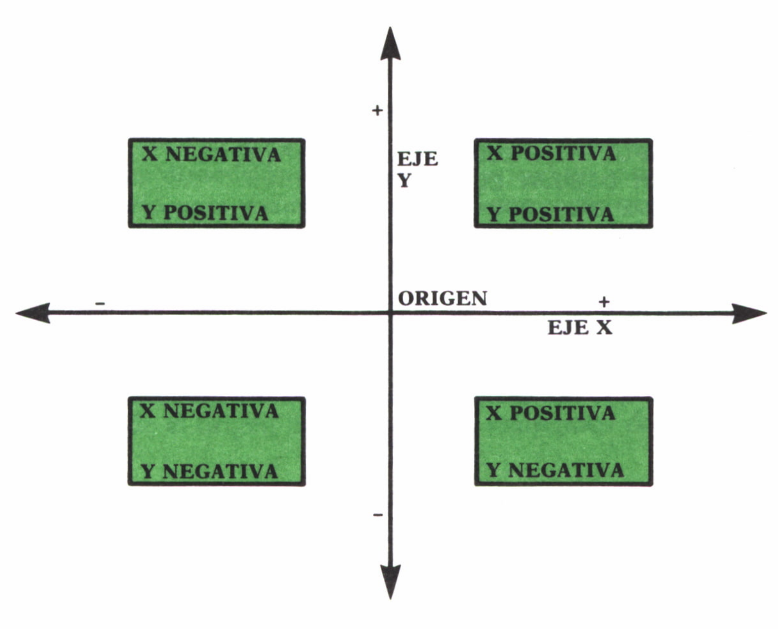


Fig. 2.3

Todo esto lo podemos ver de una forma práctica gracias al programa siguiente:

```
10 REM *******************************
20 REM **                           **
30 REM **   PROGRAMA PARA AMSTRAD   **
40 REM **                           **
50 REM **      VECTORES  2 - D      **
60 REM **                           **
70 REM *******************************
80 MODE 1
90 CLS:CLG
100 REM PINTA LOS EJES
110 MOVE 320,0
120 DRAW 320,399,1
130 MOVE 0,200
140 DRAW 639,200
150 REM ESCRIBE NOMBRE EJES
160 MOVE 330,390
170 DRAWR 5,-5:DRAWR 5,5:DRAWR-10,-10
180 MOVE 600,220
190 DRAWR 10,-10:MOVER 0,10:DRAWR -10,-10
200 REM LEER DATOS
```

```
210 LOCATE 1,25
220 PRINT SPC(35)
230 LOCATE 1,25
240 INPUT "INTRODUCIR X",Y ->";X,Y
250 REM DIBUJA EL VECTOR
260 MOVE 320,200+Y
270 DRAWR X,0,2
280 DRAWR 0,-Y
290 MOVE 320,200
300 DRAWR X,Y,3
310 REM REPETIR EL PROCESO
320 GOTO 200
```

Programa 2.1

Este programa está pensado para Amstrad, pero es muy sencillo cambiarlo para otros ordenadores, como Spectrum o IBM.

Su uso es muy sencillo; al ponerlo en marcha pedirá un par de números, las coordenadas X e Y, y una vez comprobadas que son correctas pintará el vector correspondiente a partir del centro de la pantalla.

Pero ya vimos en el capítulo anterior que las pantallas de los ordenadores tienen su origen de coordenadas en una esquina de la pantalla, por lo que el programa debe transformar el punto introducido al verdadero valor de las coordenadas. Para hacer esto realizamos los siguientes pasos:

1. Nos ponemos en el origen verdadero de la pantalla.
2. Desplazamos el punto *sin pintar* al centro de ella.
3. Pintamos de forma relativa desde donde estamos una línea con las coordenadas que hemos dado al vector.

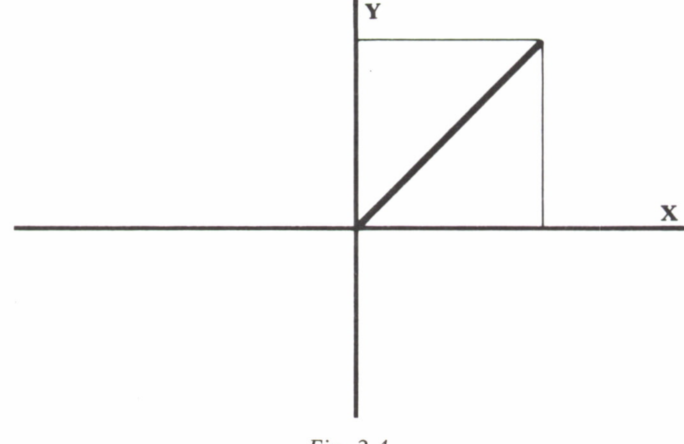


Fig. 2.4

Este tipo de representacion no es muy útil para dibujar en dos dimensiones, ya que podemos utilizar directamente las coordenadas de la pantalla y olvidarnos de vectores, ejes, etc.

Pero esto se hace imprescindible a la hora de representar cuerpos con volumen, en tres dimensiones, ya que, en principio, nuestros ordenador no está preparado para visualizar cuerpos en tres dimensiones y hay que recurrir a los vectores y coordenadas para poder hacerlo.

Al igual que en dos dimensiones, en el espacio tenemos un origen a partir del cual vamos a medir los vectores, y en vez de tener sólo dos ejes, X e Y, tenemos tres ejes que son perpendiculares entre sí, X, Y, que son los que ya conocemos, y un tercer eje, el Z, que nos va a servir para medir la profundidad, la tercera dimensión. En el eje Z se toma por convenio el signo positivo desde el origen hacia nosotros, «saliendo» de la pantalla.

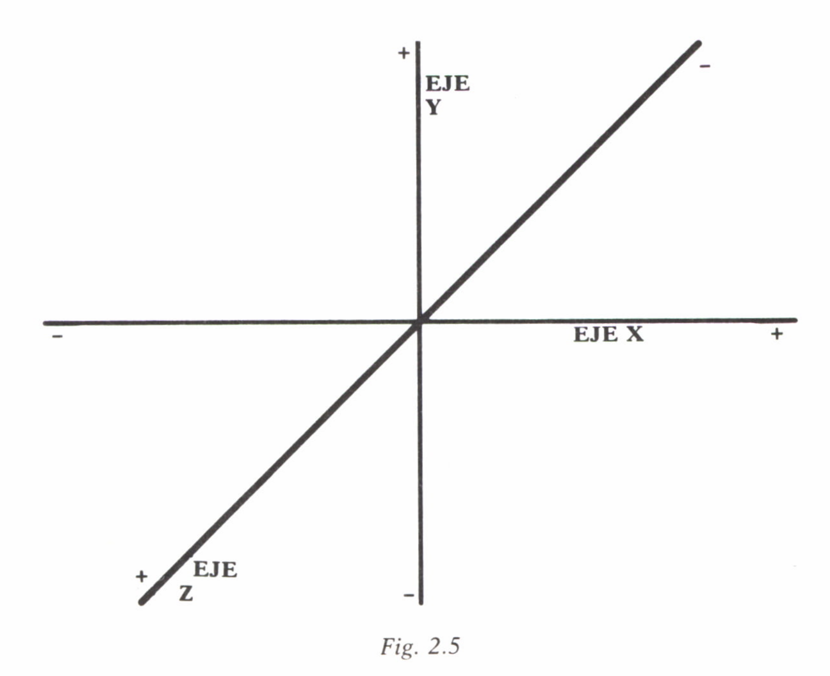


Fig. 2.5

Al tener tres ejes, un punto del espacio queda perfectamente determinado dando tres valores, las coordenadas para cada uno de los ejes. El orden en el que los vamos a escribir va a ser:

(X,Y,Z)

Para ver mejor esto podemos utilizar el programa que viene a continuación, que nos preguntará el valor de las tres coordenadas y representará el vector en la pantalla:

```
10 REM ********************************
20 REM **                            **
30 REM **   PROGRAMA PARA AMSTRAD    **
40 REM **                            **
50 REM **      VECTORES  3 - D       **
60 REM **                            **
70 REM ********************************
80 MODE 1
90 CLS:CLG
100 dx=0.5:dy=0.5
110 REM PINTA LOS EJES
120 MOVE 320,0
130 DRAW 320,399,1
140 MOVE 0,200
150 DRAW 639,200
160 MOVE 120,0
170 DRAW 520,399
180 REM ESCRIBE NOMBRE EJES
190 MOVE 330,390
200 DRAWR 5,-5:DRAWR 5,5:DRAWR-10,-10
210 MOVE 600,220
220 DRAWR 10,-10:MOVER 0,10:DRAWR -10,-10
230 MOVE 125,30
240 DRAWR 10,0:DRAWR -10,-10:DRAWR 10,0
250 REM LEER DATOS
260 LOCATE 1,25
270 PRINT SPC(35)
280 LOCATE 1,25
290 INPUT "INTRODUCIR X,Y,Z ->";X,Y,Z
300 REM DIBUJA EL VECTOR
310 MOVE 320-z*dx,200-z*dy
320 DRAWR x,0,2
330 MOVE 320+x,200
340 DRAWR -z*dx,-z*dy
350 DRAWR 0,y
360 DRAW 320,200,3
370 REM REPETIR EL PROCESO
380 GOTO 250
```

Programa 2.2

Este tipo de representación en la que a cada punto le hacemos corresponder los tres valores de las coordenadas de cada eje se llama representación «cartesiana» del punto y es la que más vamos a utilizar a lo largo del libro.

Existe otro tipo de representación que en vez de dos valores de las tres coordenadas utiliza como datos la distancia del punto al origen y dos ángulos, el formado por la recta entre el punto y el origen y el plano XZ, y el formado por la recta que va del punto al origen y el eje Z.

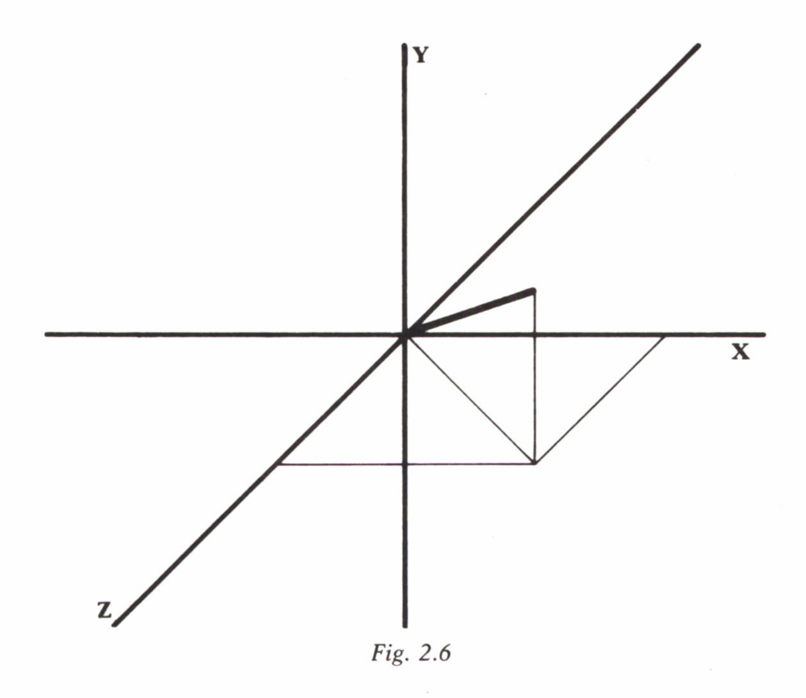


Fig. 2.6

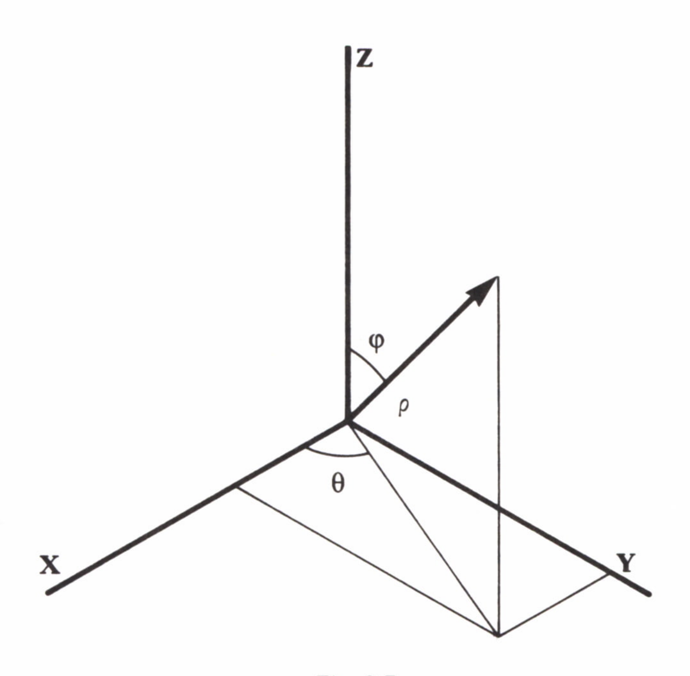


Fig. 2.7

En esta figura ρ es la distancia, θ el primer ángulo y φ el segundo.

En algunas ocasiones los ejes no estarán distribuidos como hemos dicho, sino que el eje vertical de la pantalla será el eje Z y el eje que «sale» de ella el eje Y.

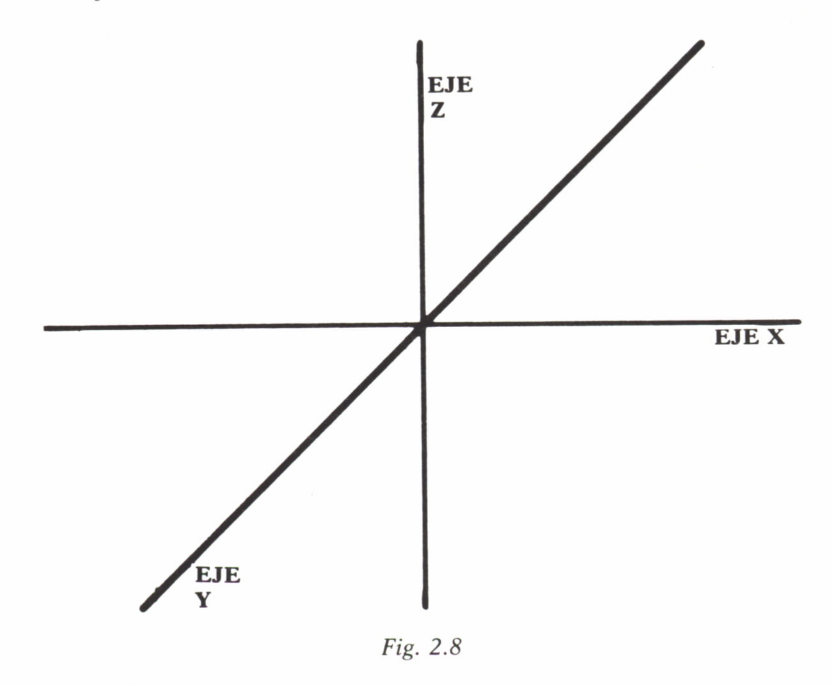


Fig. 2.8

Por último, vamos a contar las dos operaciones más comunes que se realizan con los vectores.

SUMA: Para sumar dos vectores se suman las coordenadas correspondientes, dando otro vector como resultado.

Ej.: (2,3) + (5,7) = (2+5,3+7)

MULTIPLICACION POR UN NUMERO: Multiplicar un vector por un número cualquiera consiste en multiplicar cada una de las coordenadas de ese vector por el número dando otro vector como resultado.

Ej.: 5*(6,9) = (5*6,5*9)

Dados dos puntos cualesquiera del espacio (X1,Y1.Z1) y (X2,Y2,Z2), podemos calcular el vector que les une restando las coordenadas de los dos.

(VX,VY,VZ) = (X2-X1, Y2-Y1, Z2-Z1)

REPRESENTANDO OBJETOS EN LA PANTALLA 3

INTRODUCCION

HASTA ahora no hemos dicho nada de cómo representar los puntos del espacio en la pantalla, ya que cada punto tiene tres coordenadas y hay que visualizarlas sobre un plano, esto es, sobre una superficie en dos dimensiones. El problema se puede comparar con el paso de una figura salida en el espacio al papel de un dibujante. En los programas del capítulo anterior hemos utilizado una serie de «trucos» o técnicas que nos permitían dibujar un vector en tres dimensiones sobre el plano de la pantalla.

Estas técnicas son lo que se llama comúnmente técnicas de proyección, ya que lo que hacen es «proyectar» un punto del espacio sobre un plano de la siguiente forma:

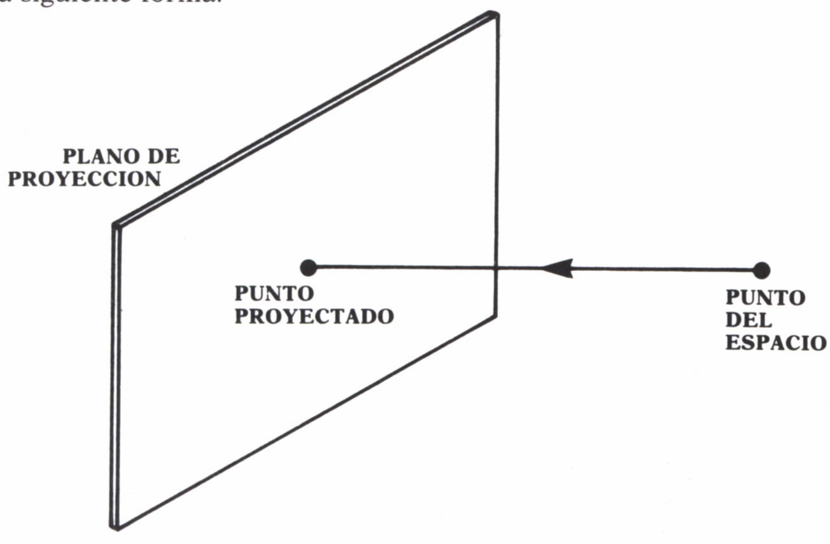


Fig. 3.1

Desde el punto del espacio se traza una recta según una dirección determinada hasta que corte al plano de proyección donde se va a dibujar la figura.

Existen diversas formas de proyectar sobre un plano, pero nosotros vamos a manejar sólo dos de ellas: la proyección paralela y la proyección cónica.



PROYECCION PARALELA

La proyección paralela es la más sencilla que podemos realizar. En este caso suponemos que el ojo del observador está lo suficientemente lejos del plano de proyección como para que las rectas que pasan por los puntos del cuerpo sean todas paralelas entre sí, o sea, que en este caso la distancia a la que está el observador no influye en la representación.

Como «plano de proyección», que en resumen es la pantalla del ordenador, vamos a utilizar en principio el plano XY con lo que todo aquello que pertenezca a este plano se dibujará tal y como es en realidad, mientras que todo lo que esté fuera de este plano sufrirá una deformación.

La forma más sencilla de proyección paralela es suponer que el observador tiene el plano XY perpendicularmente a él, de tal forma que proyectar un objeto sobre el plano consiste solamente en eliminar la coordenada Z de cada punto a pintar.

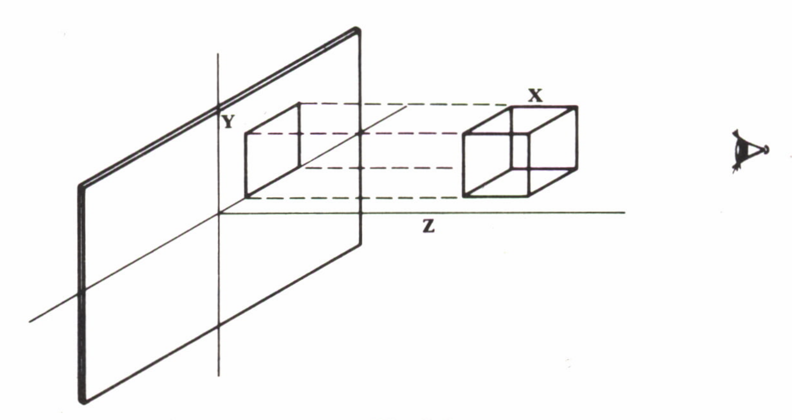


Fig. 3.2

Ya sabemos cómo proyectar un punto en la pantalla de nuestro ordenador, pero ¿cómo pintar rectas que están en el espacio? Es muy sencillo: basta con calcular las proyecciones de los extremos de la recta y trazar en la pantalla (con un comando DRAWR, por ejemplo) una línea que los una.

Si (X1,Y1) y (X2,Y2) son las proyecciones de los puntos, las instrucciones serían:

PLOT X1,Y1

DRAWR X2-X1,Y2-Y1

Este es el método general, independientemente del tipo de proyección que emplearemos para trazar cuerpos en la pantalla de nuestro ordenador.

1. Calculamos la proyección de un punto sobre el plano XY.
2. Nos movemos a él o lo pintamos.
3. Calculamos la proyección del siguiente punto a pintar y trazamos una recta en la pantalla del ordenador entre los dos puntos que hemos calculado.

Lo único que variará de este método será el sistema que utilicemos para calcular la proyección del punto.

El siguiente programa utiliza una proyección paralela tal y como la hemos descrito y nos pide dos puntos, trazando la proyección de la recta que los une.

```
 10 REM ****************************
 20 REM **                        **
 30 REM ** PROGRAMA PARA SPECTRUM **
 40 REM **                        **
 50 REM ** PROYECCION ORTOGONAL   **
 60 REM **                        **
 70 REM ****************************
 80 CLS
 90 REM PINTA LOS EJES
100 PLOT 127,0
110 DRAW 0,174
120 PLOT 0,83
130 DRAW 255,0
140 REM PREGUNTA LOS DATOS
150  INPUT "PRIMER PUNTO ";xin,yin,zin
160 INPUT "SEGUNDO PUNTO ";xf,yf,zf
170 REM CALCULA LA PROYECCION
180 LET x1=xin: LET y1=yin
190 LET x2=xf: LET y2=yf
200 REM CENTRA EL ORIGEN
210 LET x1=x1+127: LET y1=y1+83
220 LET x2=x2+127: LET y2=y2+83
230 REM DIBUJA
240 PLOT x1,y1
250 DRAW x2-x1,y2-y1
260 REM REPITE EL PROCESO
270 GO TO 140
```

Programa 3.1

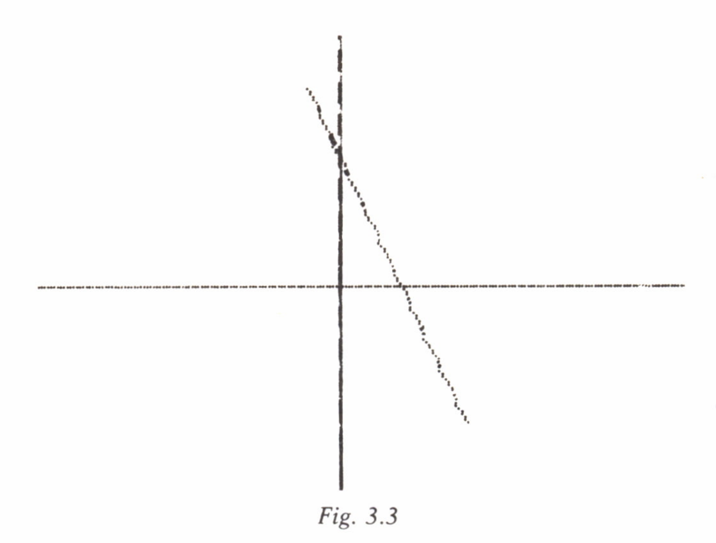


Fig. 3.3

Como vemos en el dibujo que produce este programa, el eje Z, perpendicular a la pantalla, no se ve, como consecuencia del tipo de proyección que utilizamos.

Si queremos que el programa pinte una figura un poco más compleja sin necesidad de meter los datos cada vez, podemos meter las coordenadas de cada punto en una sentencia DATA e ir leyendo los puntos y proyectando. Como una figura compleja puede necesitar que nos movamos sin pintar, para cada punto añadiremos un dato más que nos indicará si la línea hay que pintarla o sólo moverse sin pintar, también nos va a servir para indicar si hay más datos o hemos llegado al final de la figura. El formato es:

MODO	
MOVER 0	X Y Z
PINTAR 1	
FIN -1	

De esta forma, en el programa siguiente pintamos un cubo en la pantalla.

```
10 REM ****************************************
20 REM **                                    **
30 REM **       VERSION PARA IBM             **
40 REM **                                    **
50 REM **       PROYECCION ORTOGONAL         **
60 REM **                                    **
70 REM ****************************************
```

```
80 SCREEN 1
90 CLS
100 XO=160:YO=100
110 REM VER SI HA ACABADO
120 READ COLORP
130 IF COLORP=-1 THEN GOTO 400
140 REM LEER EL PUNTO
150 READ X,Y,Z
160 REM CALCULAR LA PROYECCION
170 PX = X
180 PY = Y
190 REM TRASLADAR EL ORIGEN
200 X1=PX+XO
210 Y1=PY+YO
220 REM VER SI HAY QUE DIBUJAR
230 IF COLORP= 0 THEN PRESET (X1,Y1)
240 IF COLORP= 1 THEN LINE - (X1,Y1)
250 REM REPETIR EL PROCESO
260 GOTO 120
300 DATA 0,0,0,0,1,50,0,0,1,50,50,0,1,0,50,0
310 DATA 1,0,50,50,1,0,0,50,1,50,0,50
320 DATA 1,50,50,50,1,0,50,50,0,50,50,50
330 DATA 1,50,50,0,0,50,0,50,1,50,0,0
340 DATA 0,0,0,50,1,0,0,0,1,0,50,0,-1
400 GOTO 400

VARIACIONES PARA M.S.X.

LINEA  80 SCREEN 2
```

Programa 3.2

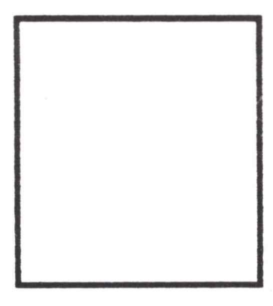


Fig. 3.4

El resultado de este programa es un poco desalentador, ya que sólo produce un cuadrado en la pantalla. Esto es por el tipo de proyección empleado, al ser una proyección perpendicular al plano XY, y estar el cubo pues-

to de tal forma que las proyecciones de los vértices coinciden en un solo punto.

Como vemos, la proyección paralela ortogonal es muy sencilla, pero no permite hacer gráficos muy sofisticados.

Por ello, dentro de las proyecciones paralelas, podemos utilizar otro método, que es el de la proyección paralela oblicua. Este método consiste en suponer que el ojo no está en una dirección perpendicular al plano XY, sino que está «torcido», inclinado respecto a él.

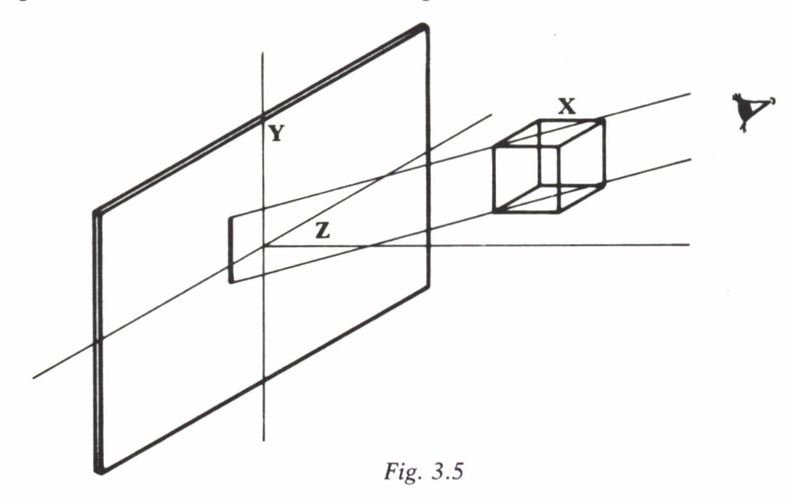


Fig. 3.5

En este caso la fórmula que nos da el valor del punto sobre el plano XY es un poco más complicada.

Primero tenemos que establecer la dirección desde donde vamos a proyectar el punto. Esto lo hacemos dando las tres coordenadas de un punto cualquiera del espacio cuyo vector de posición desde el origen tenga la misma dirección que la de proyección.

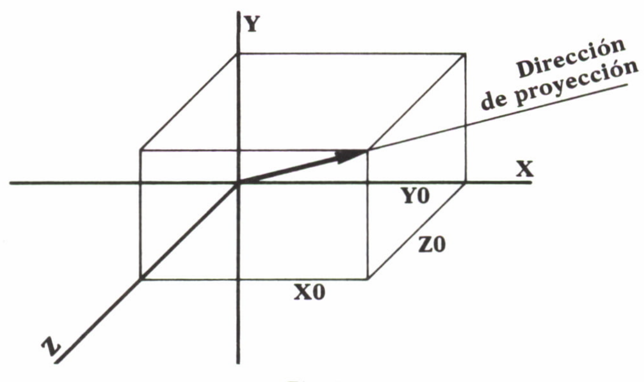


Fig. 3.6

Ahora utilizando un razonamiento geométrico sencillo, podemos deducir que la proyección de una coordenada X sobre el plano XY es:

$$PX = X - \frac{Z*X0}{Z0}$$

Siendo X,Y,Z el punto a proyectar y X0,Y0,Z0 las coordenadas del punto que nos da la dirección de proyección.

En la figura siguiente podemos ver que es muy fácil de calcular por semejanza de triángulos.

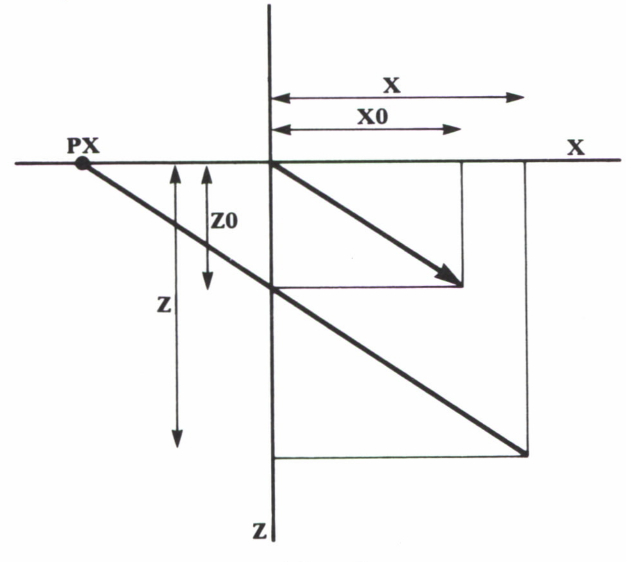


Fig. 3.7

Igualmente para la coordenada Y queda:

$$PY = Y - \frac{Z*Y0}{Z0}$$

Modificando ligeramente el programa de dibujo del cubo para poder emplear esta proyección, nos queda este programa:

```
10 REM ********************************
20 REM **                            **
30 REM **   PROGRAMA PARA AMSTRAD    **
40 REM **                            **
50 REM **      PROYEC. OBLICUA       **
60 REM **                            **
70 REM ********************************
```

```
80 MODE 1
90 CLS:CLG
100 X0=320:Y0=200
105 LOCATE 1,25
106 INPUT "LINEA DE PROYECCION X,Y,Z";XP,YP,ZP
110 REM LEER SI SE ACABA
120 READ COLOR
130 IF COLOR = -1 THEN GOTO 320
140 REM LEER EL PUNTO
150 READ X,Y,Z
160 REM CALCULAR LA PROYECCION
170 X1=X-(Z*XP)/ZP
180 Y1=Y-(Z*YP)/ZP
190 REM TRASLADAR EL ORIGEN
200 X1=X1+X0
210 Y1=Y1+Y0
220 REM VER SI HAY QUE DIBUJAR
230 IF COLOR = 0 THEN MOVE X1,Y1
240 IF COLOR = 1 THEN DRAW X1,Y1
250 REM REPETIR EL PROCESO
260 GOTO 120
270 DATA 0,0,0,0,1,100,0,0,1,100,100,0,1,0,100,0
280 DATA 1,0,100,100
290 DATA 1,0,0,100,1,100,0,100,1,100,100,100
300 DATA 1,0,100,100,0,100,100,100,1,100,100,0
310 DATA 0,100,0,100,1,100,0,0,0,0,0,100,1,0,0,0,1,0,
    100,0,-1
320 GOTO 320
```

Programa 3.3

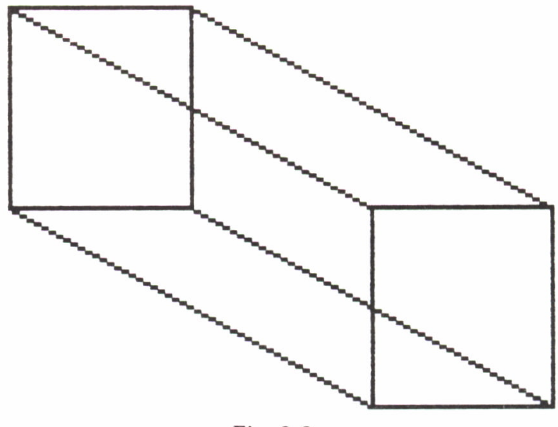


Fig. 3.8

Si queremos cambiar la dirección de proyección basta con cambiar los valores de X0,Y0 y Z0 en el programa.

Como vemos, ya da una representación del cubo mucho más realista y vistosa.

Si en este programa cambiamos las sentencias DATA por otras podremos representar cualquier objeto que queramos. A continuación damos algunos ejemplos:

```
10 REM ****************************
20 REM **                        **
30 REM ** PROGRAMA PARA SPECTRUM **
40 REM **                        **
50 REM **   PROYECCION OBLICUA   **
60 REM **                        **
70 REM ****************************
80 CLS
90 LET xo=127: LET yo=83
100 REM LEER LOS DATOS
110 INPUT "LINEA DE PROYECCION ";xp,yp,zp
120 REM LEER SI SE ACABA
130 READ color
140 IF color=-1 THEN GO TO 500
150 REM LEER EL PUNTO
170 READ x,y,z
180 REM CALCULA LA PROYECCION
190 LET x1=x-(z*xp)/zp
200 LET y1=y-(z*yp)/zp
210 REM TRASLADA EL ORIGEN
220 LET x1=x1+xo
230 LET y1=y1+yo
240 REM VER SI HAY QUE DIBUJAR
250 IF color=1 THEN PLOT x1,y1
260 IF color=0 THEN DRAW x1-ux,y1-uy
265 LET ux=x1:  LET uy=y1
270 REM REPETIR EL PROCESO
280 GO TO 120
300 DATA 0,0,0,0,1,30,0,0,1,30,0,30,1,0,0,30
310 DATA 1,0,0,0,1,15,30,15,1,0,0,30,0,30,0,30
320 DATA 1,15,30,15,1,30,0,0,-1
```

Programa 3.4

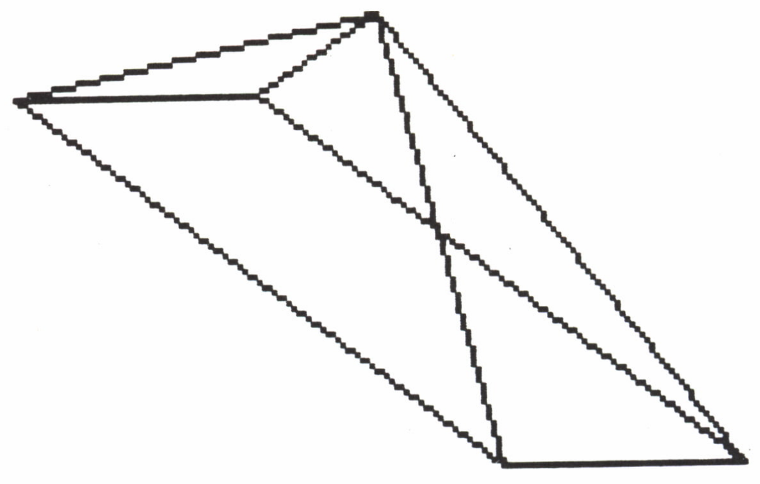


Fig. 3.9

```
10 REM ***************************************
20 REM **                                   **
30 REM **      VERSION PARA IBM             **
40 REM **                                   **
50 REM **      PROYECCION OBLICUA           **
60 REM **                                   **
70 REM ***************************************
80 SCREEN 1
90 CLS
100 XO=160:YO=100
110 LOCATE 25,1
120 INPUT "LINEA DE PROYECCION :";XP,YP,ZP
130 CLS
140 REM VER SI HA ACABADO
150 READ COLORP
160 IF COLORP=-1 THEN GOTO 350
170 REM LEER EL PUNTO
180 READ X,Y,Z
190 REM CALCULAR LA PROYECCION
200 PX = X-(Z*XP)/ZP
210 PY = Y-(Z*YP)/ZP
220 REM TRASLADAR EL ORIGEN
230 X1=PX+XO
240 Y1=PY+YO
250 REM VER SI HAY QUE DIBUJAR
260 IF COLORP= 0 THEN PRESET (X1,Y1)
270 IF COLORP= 1 THEN LINE -(X1,Y1)
280 REM REPETIR EL PROCESO
```

```
290 GOTO 150
300 DATA 0,0,0,0,1,50,0,0,1,50,0,50,1,0,0,50
310 DATA 1,0,0,0,1,25,50,25,1,0,0,50,0,50,0,50
320 DATA 1,25,50,25,1,50,0,0,1,25,-50,25,1,0,0,0
330 DATA 1,25,-50,25,0,0,0,50,1,25,-50,25
340 DATA 1,50,0,50,-1
350 GOTO 350

VARIACIONES PARA M.S.X.

LINEA  80 CLS
LINEA  90 LOCATE 20,1:INPUT "LINEA DE PROYE
CCION
:";XP,YP,ZP
LINEA 110 SCREEN 2

QUITAR LAS LINEAS 120 Y 130
```

Programa 3.5

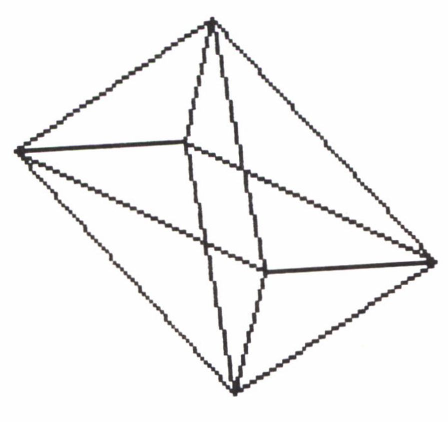


Fig. 3.10

```
10 REM ******************************
20 REM **                          **
30 REM **   PROGRAMA PARA AMSTRAD  **
40 REM **                          **
50 REM **     PROYEC. CONICA       **
60 REM **                          **
70 REM ******************************
```

```
80 MODE 1
90 CLS:CLG
100 X0=320:Y0=200
110 LOCATE 1,25
120 INPUT "DISTANCIA FOCAL,pos xy ->";D,ox,oy
130 REM LEER SI SE ACABA
140 READ COLOR
150 IF COLOR = -1 THEN GOTO 340
160 REM LEER EL PUNTO
170 READ X,Y,Z
180 REM CALCULAR LA PROYECCION
190 X1=D*(X-ox)/(D-Z)+ox
200 Y1=D*(Y-oy)/(D-Z)+oy
210 REM TRASLADAR EL ORIGEN
220 X1=X1+X0
230 Y1=Y1+Y0
240 REM VER SI HAY QUE DIBUJAR
250 IF COLOR = 0 THEN MOVE X1,Y1
260 IF COLOR = 1 THEN DRAW X1,Y1
270 REM REPETIR EL PROCESO
280 GOTO 140
290 DATA 0,0,0,100 ,1,0,10,25,1,300,0,0,0,0,10,25
300 DATA 1,0,40,0,1,0,83,0,1,33,83,0,1,65,35,0
310 DATA 0,0,40,0,1,300,0,0,0,150,4,0,1,30,0,-100
315 DATA 1,0,0,-100,1,0,10,-25,1,300,0,0,0,0,10,-25
320 DATA 1,0,40,0
330 DATA 0,0,0,-100,1,0,0,100,1,30,0,100,1,150,4,0,-1
340  GOTO 340
```

Programa 3.6

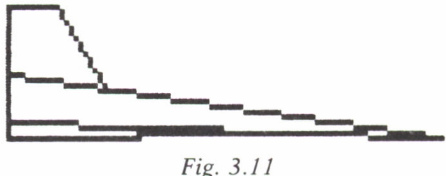


Fig. 3.11

La mejor forma de diseñar nuevos objetos consiste en dibujar sobre un papel cuadriculado la figura, con unos ejes sobre los que medir las coordenadas de cada vértice y hacer una lista de vértices de tal forma que recorramos la figura entera para poder pintarla. Si llegamos a un punto donde tenemos que movernos sin pintar, para continuar la figura en otro lado, simplemente especificamos como primer parámetro del punto el 0. Y si hemos terminado añadimos al final de los datos un −1. Hay que recordar que cada punto tiene cuatro parámetros, el primero indica si se pinta o no y los tres siguientes especifican las tres coordenadas del vértice.

Por ejemplo, para dibujar una pirámide hacemos

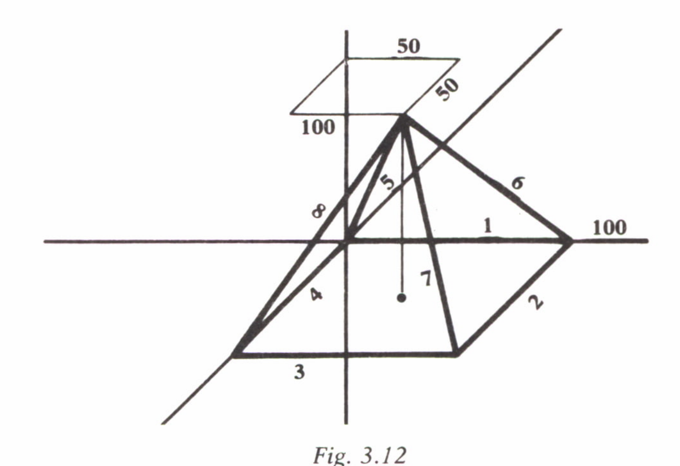


Fig. 3.12

Y la tabla queda:

Lado	Color	X	Y	Z
	0	0	0	0
1	1	100	0	0
2	1	100	0	100
3	1 1	0	0	100
4	1	0	0	0
5	1	50	100	50
6	1	100	0	0
	0	100	0	100
7	1	50	100	50
8	1	0	0	100
	–1	(FIN)		

Esta tabla luego la metemos en las **DATA** al final del programa, poniendo siempre el –1 al final, leyendo la tabla de izquierda a derecha y de arriba abajo.



PROYECCION CONICA

El sistema de proyección que hemos utilizado hasta ahora carece de algunas de las propiedades importantes de la visión real; por ejemplo, no hace más pequeñas las cosas según están más lejos del observador, por su-

poner que el ojo está a una distancia indeterminada del plano donde proyectamos. Para conseguir este efecto y, por tanto, escenas más reales emplearemos la llamada *proyección cónica.*

Este tipo de proyección se basa en suponer que el ojo está a una distancia determinada, D, del plano de proyección, y a partir de este punto trazar rayas que pasen por los vértices del objeto hasta que corten al plano de proyección donde nos darán las coordenadas del punto a pintar en la pantalla del ordenador.

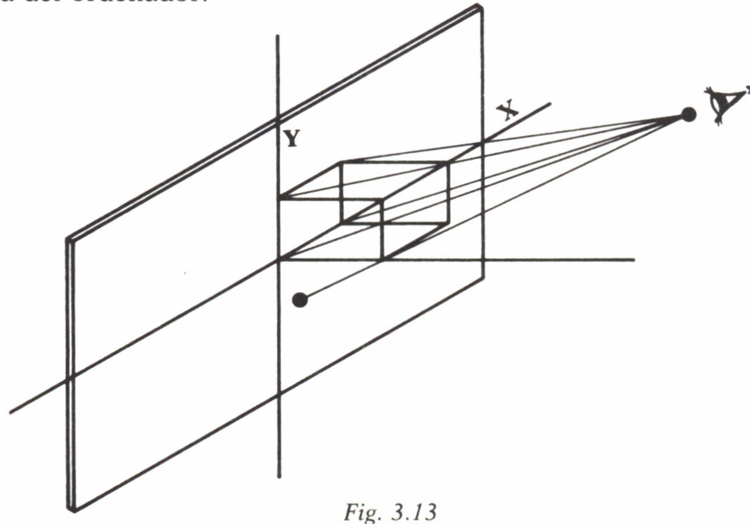


Fig. 3.13

La forma de proyección cónica más sencilla es la que el punto de vista está sobre el eje Z a una distancia determinada, y proyectamos sobre el plano XY.

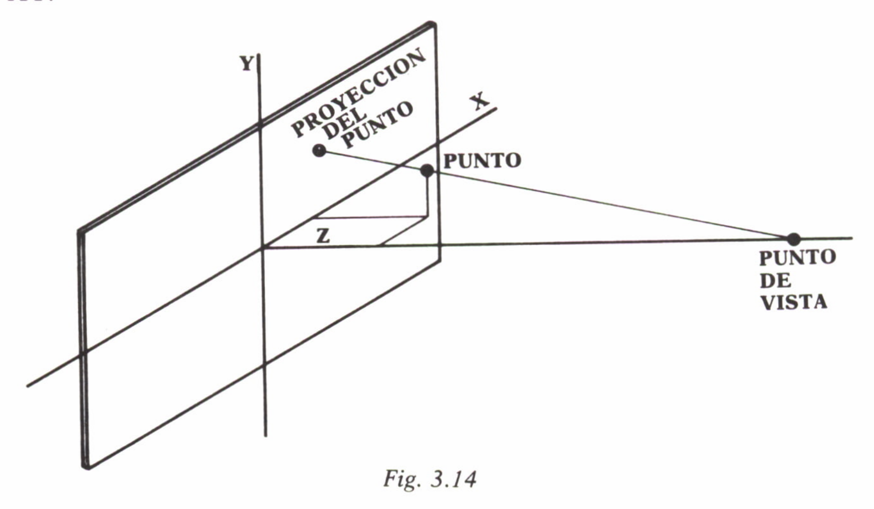


Fig. 3.14

Para calcular en este caso las fórmulas de proyección utilizaremos un razonamiento parecido al que hicimos en el caso de la proyección paralela oblicua. Si nos fijamos en la figura

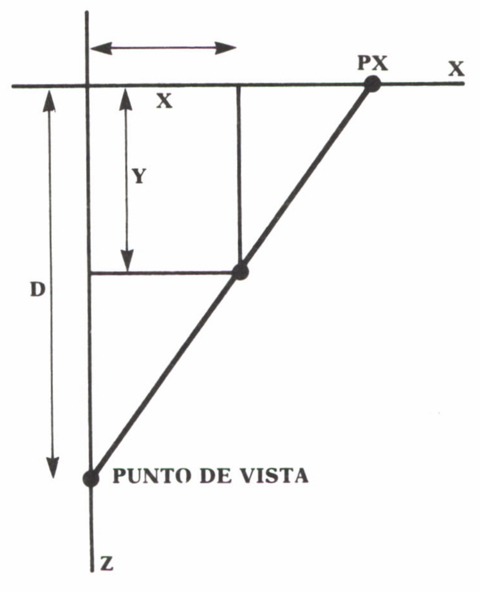


Fig. 3.15

y aplicando semejanza de triángulos obtenemos la fórmula para la proyección de la coordenada X:

$$PX = \frac{D}{D - Z} \cdot X$$

donde D es la distancia del punto de vista al origen en el eje Z, Z y X son los valores de las coordenadas del punto a proyectar y PX es el valor de la coordenada X ya proyectada sobre el plano XY.

Igualmente, para calcular la coordenada Y de la proyección tenemos:

$$PY = \frac{D}{D - Z} \cdot Y$$

Por supuesto, una vez calculados los valores de las coordenadas de la proyección, tenemos que convertirlos a verdaderos valores sobre la pantalla de nuestro ordenador, trasladando al centro de la pantalla el origen de coordenadas.

El programa que tenemos a continuación es un ejemplo de proyección cónica dibujando en la pantalla un cubo en perspectiva.

```
10 REM ****************************************
20 REM **                                    **
30 REM **      VERSION PARA IBM              **
40 REM **                                    **
50 REM **      PROYECCION CONICA             **
60 REM **                                    **
70 REM ****************************************
80 SCREEN 1
90 CLS
100 XO=160:YO=100
105 REM  leer datos
110 LOCATE 24,1
120 INPUT " Distancia focal -> ";D
125 CLS
130 REM VER SI HA ACABADO
140 READ COLORP
150 IF COLORP=-1 THEN GOTO 340
160 REM LEER EL PUNTO
170 READ X,Y,Z
180 REM CALCULAR LA PROYECCION
190 PX = D*X/(D-Z)
200 PY = D*Y/(D-Z)
210 REM TRASLADAR EL ORIGEN
220 X1=PX+XO
230 Y1=PY+YO
240 REM VER SI HAY QUE DIBUJAR
250 IF COLORP= 0 THEN PRESET (X1,Y1)
260 IF COLORP= 1 THEN LINE - (X1,Y1)
270 REM REPETIR EL PROCESO
280 GOTO 140
290 DATA 0,0,0,0,1,50,0,0,1,50,50,0,1,0,50,0
300 DATA 1,0,50,50,1,0,0,50,1,50,0,50
310 DATA 1,50,50,50,1,0,50,50,0,50,50,50
320 DATA 1,50,50,0,0,50,0,50,1,50,0,0
330 DATA 0,0,0,50,1,0,0,0,1,0,50,0,-1
340 GOTO 340

VARIACIONES PARA M.S.X.

LINEA  80 CLS
LINEA  90 LOCATE 20,1
LINEA 100 INPUT "DISTANCIA FOCAL -> ";D
LINEA 110 SCREEN 2
LINEA 120 XO=160:YO=100

QUITAR LA LINEA 125
```

Programa 3.7

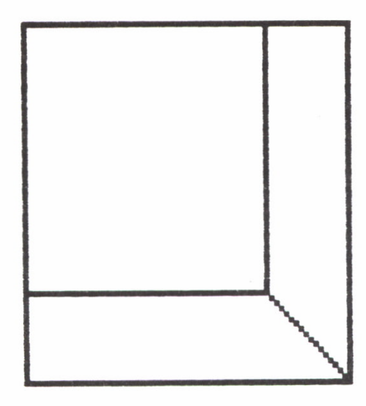


Fig. 3.16

En este tipo de proyección también podemos desplazar el punto de vista de tal manera que la proyección resultante sea oblicua, es decir, en vez de que el ojo esté situado sobre el eje Z pueda estar situado en cualquier punto del espacio.

En este caso las fórmulas de proyección que resultan son:

$$PX = \frac{D}{D - Z} * (X-0X) + 0X$$

y

$$PY = \frac{D}{D-Z} * (Y-0Y) + 0Y$$

donde X,Y,Z son las coordenadas espaciales del punto; PX,PY, las coordenadas del punto proyectado; D, la distancia perpendicular desde el punto de vista al plano XY; y OX,OY, las coordenadas X e Y del punto de vista.

En esta proyección cónica oblicua el observador sigue mirando al plano XY de forma perpendicular, pero está desplazado respecto al eje Z las cantidades OX y OY.

Un ejemplo de esta proyección lo da el programa siguiente:

```
10 REM ****************************
20 REM **                        **
30 REM ** PROGRAMA PARA SPECTRUM **
40 REM **                        **
50 REM **   PROYECCION CONICA    **
60 REM **                        **
70 REM ****************************
80 CLS
90 LET xo=127: LET yo=83
```

```
100 REM LEER LOS DATOS
110 INPUT "DISTANCIA FOCAL,X,Y ";d,ox,oy
120 REM LEER SI SE ACABA
130 READ color
140 IF color=-1 THEN GO TO 500
150 REM LEER EL PUNTO
170 READ x,y,z
180 REM CALCULA LA PROYECCION
190 LET x1=d*(x-ox)/(d-z)+ox
200 LET y1=d*(y-oy)/(d-z)+oy
210 REM TRASLADA EL ORIGEN
220 LET x1=x1+xo
230 LET y1=y1+yo
240 REM VER SI HAY QUE DIBUJAR
250 IF color=1 THEN PLOT x1,y1
260 IF color=0 THEN DRAW x1-ux,y1-uy
265 LET ux=x1: LET uy=y1
270 REM REPETIR EL PROCESO
280 GO TO 120
300 DATA 0,0,0,0,1,30,0,0,1,30,30,0
310 DATA 1,0,30,0,1,0,30,30
320 DATA 1,0,0,30,1,30,0,30,1,30,30,30
330 DATA 1,0,30,30,1,30,30,30,1,30,30,0
340 DATA 0,30,0,30,1,30,0,0,0,0,0,30
350 DATA 1,0,0,0,1,0,30,0,-1
```

Programa 3.8

Podemos cambiar en él las coordenadas del punto de vista para obtener diversas vistas del objeto, cambiando los valores de D, OX y OY.



ESTRUCTURA GENERAL DE LOS PROGRAMAS

Si nos fijamos en los programas que hemos realizado en este capítulo, todos tienen la misma estructura:

```
BORRADO DE LA PANTALLA
INICIALIZACION DE VARIABLES
LEER UN PUNTO
REPITE HASTA COLOR = -1
    CALCULA PROYECCIONES (PX,PY)
    DESPLAZA EL ORIGEN
    PINTA UNA LINEA O MUEVE A ESE PUNTO
    LEE EL SIGUIENTE PUNTO
FIN DEL REPITE
FIN DEL PROGRAMA
DATOS DEL OBJETO A REPRESENTAR
```

Esta estructura general de los programas se puede utilizar para cualquier tipo de proyección cambiando sólo dos partes que varían según sea el tipo de proyección; primero cambiamos la inicialización de variables para meter en ella todos los valores que necesitamos en cada caso, y acto seguido cambiamos la parte de cálculo de proyecciones para ajustarlas al tipo escogido por nosotros.

Es una estructura muy sencilla y se puede aplicar en todos los casos, incluso algunos más complejos que veremos más adelante. Podemos incluso investigar por nuestra cuenta y encontrar nuevas fórmulas de proyección con las que mejorar los dibujos de cuerpos en tres dimensiones.



GENERALIZACION DEL SISTEMA DE PROYECCION

Ya conocemos los sistemas básicos para representar en la pantalla del ordenador cuerpos en tres dimensiones. Podemos con ellos representar un cuerpo desde cualquier vista que nos imaginemos, pero con una limitación; en todos los métodos que utilizamos hasta ahora hemos proyectado sobre el plano XY, y esto impide que veamos los cuerpos desde otra vista que no sea la frontal, esto es, el plano XY.

Para solventar este problema debemos de variar ligeramente el enfoque de la solución para poder proyectar sobre cualquier plano, lo que nos permitirá poder ver el objeto desde cualquier ángulo y posición del espacio. Gracias a esto podemos evitar el realizar proyecciones oblicuas, ya que para ver el objeto desde otra perspectiva lo que haremos será girar el plano de proyección, y no solamente la posición del observador, con lo que éste siempre está orientado perpendicularmente a dicho plano.

Vamos a profundizar un poco más; hemos dicho que giramos el plano de proyección, como se ve en la siguiente figura:

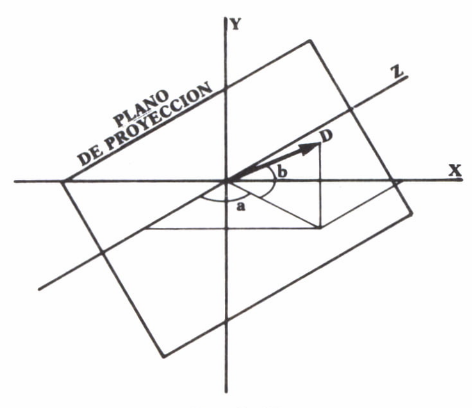


Fig. 3.17

Este plano lo vamos a especificar en coordenadas esféricas con un vector perpendicular a él cuyos datos serán:

= D - Distancia del observador al plano de proyección.
= a - ángulo que forma el observador con el plano YZ.
= b - ángulo que forma el observador con el plano ZX.

Esta solución puede parecer un poco extraña para los que no tengan unos conocimientos de geometría vectorial, pero un plano queda perfectamente especificado dando un vector perpendicular a él. De todas formas, si esto o los procesos matemáticos siguientes no son entendidos, no es ningún obstáculo para poder realizar el programa de forma correcta, como ya veremos.

El paso siguiente es un poco más complicado, y consiste en expresar los vectores de cada uno de los vértices en las coordenadas referidas al plano de proyección. Resumiendo, el proceso matemático, para aquellos que quieran investigar por su cuenta, consiste en:

= Establecer un nuevo sistema de referencia girando los vectores unitarios de los X,Y,Z, según los ángulos a y b, para obtener un nuevo sistema de referencia en el cual expresar los vectores de posición de los vértices.

Pero podemos olvidarnos de esta definición matemática tan estricta considerando que, en el fondo, lo que hacemos es transformar la figura de forma que quede tal y como la veríamos desde el punto de vista que estemos situados.

Las fórmulas que resultan de aplicar esta transformación son las siguientes:

XT = X*COS(a) - Z*SIN(a)

YT = Y*COS(b) - Z*COS(a)*SIN(b) X*SIN(a)*SIN(b)

ZT = Z*COS(a)*COS(b) + X*SIN(a)*COS(b) + Y*SIN(b)

Una vez obtenidos los puntos transformados de los vértices, lo único que tenemos que hacer es proyectar según alguna de las fórmulas que conocemos estos puntos, olvidándonos que el plano de proyección está girado, ya que con la transformación anterior es como si este plano se hubiera convertido en el plano XY normal.

Con este método podemos ver ya desde cualquier punto de vista nuestro objeto, e incluso podríamos hacer un programa que cambiara los ángulos de observación para ir viendo el objeto desde perspectivas distintas de una forma continua, dando así la impresión de que estamos «volando» sobre él. Este es el método en el que se harán los famosos «simuladores de vuelo» para microordenadores.

Empleando todos estos conceptos en forma de sentencias obtenemos un programa como el siguiente:

```
10 REM ******************************
20 REM **                          **
30 REM **   PROGRAMA PARA AMSTRAD  **
40 REM **                          **
50 REM **     PROYEC. GENERAL      **
60 REM **                          **
70 REM ******************************
80 MODE 1
90 CLS:CLG
100 X0=320:Y0=200:d=200
110 LOCATE 1,25
120 INPUT "angulos A,B->";A,B
130 A=A*PI/180:B=B*PI/180
140 A1=COS(A):A2=SIN(A)
150 B1=COS(B):B2=SIN(B)
160 REM LEER SI SE ACABA
170 READ COLOR
180 IF COLOR = -1 THEN GOTO 410
190 REM LEER EL PUNTO
200 READ X,Y,Z
210 REM CALCULAR LA PROYECCION
220 X1=A1*X-A2*Z
230 Y1=Y*B1-Z*A1*B2-X*A2*B2
240 Z1=Z*A1*B1+X*A2*B1+Y*B2
250 X1=D*X1/(D-Z1)
260 Y1=D*Y1/(D-Z1)
270 REM TRASLADAR EL ORIGEN
280 X1=X1+X0
290 Y1=Y1+Y0
300 REM VER SI HAY QUE DIBUJAR
310 IF COLOR = 0 THEN MOVE X1,Y1
320 IF COLOR = 1 THEN DRAW X1,Y1
330 REM REPETIR EL PROCESO
340 GOTO 170
350 DATA 0,0,0,100 ,1,0,10,25,1,300,0,0,0,0,10,25
360 DATA 1,0,40,0,1,0,83,0,1,33,83,0,1,65,35,0
370 DATA 0,0,40,0,1,300,0,0,0,150,4,0,1,30,0,-100
380 DATA  1,0,0,-100,1,0,10,-25,1,300,0,0,0,0,10,-25
390 DATA 1,0,40,0
400 DATA 0,0,0,-100,1,0,0,100,1,30,0,100,1,150,4,0,-1
410 GOTO 410
```

Programa 3.9

Como vemos, la estructura es muy parecida a la planteada en el apartado anterior, añadiendo antes del cálculo de las proyecciones las fórmu-

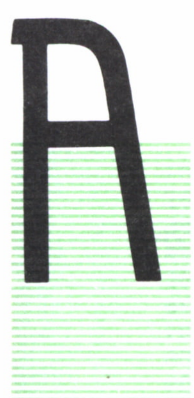
las para transformar los puntos, y en la parte de inicialización de variables unos cálculos previos necesarios para realizarla. Cambiando D, a, o b en el programa conseguiremos vistas distintas del mismo cuerpo.

Con todos estos algoritmos que hemos escrito somos capaces ya de realizar un dibujo tridimensional en la pantalla tan complicado como queramos, y verlo desde cualquier punto del espacio. Hay una sofisticación más, que sería hacer que el plano de proyección no estuviera centrado en el origen, sino que podría estar situado en cualquier punto del espacio, y la modificación del algoritmo consiste en restar a cada punto de la figura el vector que nos indica el punto del espacio donde se encuentra el plano, antes de hacer las transformaciones y proyecciones de dichos puntos.

Otra sofisticación sería añadir un tercer giro de balanceo sobre el eje de observación, algo así como el alabeo de un avión.

Con estas dos modificaciones añadidas podríamos realizar un simulador de vuelo muy sencillo que nos permitiría recorrer un mundo simulado en nuestro ordenador.

GRAFICAS E HISTOGRAMAS 4



HORA trataremos un tema diferente: cómo representar datos en dos y tres dimensiones.

Uno de los principales usos del ordenador es el tratamiento de grandes cantidades de datos y la generación de nuevos datos a partir de éstos.

Una vez realizado el tratamiento de los datos tenemos grandes listas interminables de números, que son difícilmente interpretables por seres humanos.

Para llegar a una conclusión práctica dichos datos se pueden presentar bajo la forma de gráficas y de histogramas.



GRAFICAS

Esta forma suele usarse cuando existen grandes cantidades de datos y se requiere un alto grado de exactitud. Sus principales aplicaciones son científicas, tales como representación de funciones matemáticas, trayectorias de partículas, etc.

Comenzaremos con dos dimensiones.

En dos dimensiones podemos representar sólo dos datos, es decir, a cada valor en horizontal le corresponderá un único en vertical (funciones de una sola variable).

La representación se hará mediante puntos, pero podría realizarse utilizando pequeñas rectas que unan los puntos dando una mayor aproximación, cambiando la sentencia:

```
PLOT X,Y
```

por la sentencia:

```
DRAW X,Y
```

Obsérvese que la primera parte del programa desplaza el origen de coordenadas, y cambia la escala para poder apreciar adecuadamente la zona que más nos interesa.

En el siguiente ejemplo calcularemos la gráfica de la función seno.

Para representar otra función, tanto en éste como en los siguientes programas, bastará cambiar de definición de función.

```
 10 REM ****************************
 20 REM **                        **
 30 REM **PROGRAMA PARA SPECTRUM **
 40 REM **                        **
 60 REM **  GRAFICAS EN DOS - D   **
 70 REM **                        **
 80 REM ****************************
100 REM INICIALIZA VARIABLES
110 LET XM=-4: LET XX=4: LET X0=50
120 LET YM=-4: LET YX=4: LET Y0=0
130 LET XP=(XX-XM)/200: LET XSCALL=1/XP
140 LET YSCALL=200/(YX-YM)
150 FOR X=XM TO XX STEP XP
160 GO SUB 500
170 LET X1=INT ((X-XM)*XSCALL)+X0
190 LET Y1=INT ((Y-YM)*YSCALL)+Y0
200 PLOT X1,Y1
210 NEXT X
220 STOP
500 REM CALCULO DE LA FUNCION
510 LET Y=SIN (X)
520 RETURN
```

Programa 4.1

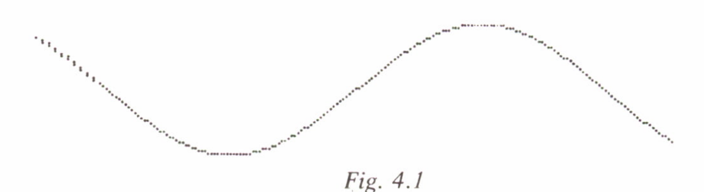


Fig. 4.1

Si queremos observar tres datos relacionados entre sí (funciones de dos variables), en los que la altura depende de la posición en un plano, debemos usar tres dimensiones.

Como la pantalla del ordenador sólo tiene dos, debemos proyectar sobre ella, tal como se vio en el capítulo anterior.

El siguiente programa recorre los puntos de un plano, calcula el valor a representar y lo proyecta sobre un plano. Utilizaremos la proyección obli-

cua ortogonal, pero si se quisiera utilizar otra sería suficiente con sustituir las fórmulas de la proyección por otras de las vistas en el capítulo anterior.

Al proyectar, algunos puntos pueden coincidir sobre el plano de proyección. Como sólo se deberá pintar lo que esté más cerca del observador, los de atrás quedarán ocultos (para una mayor información véase el capítulo 7). El algoritmo utilizado consiste en borrar desde el punto que dibujamos hasta el final de la pantalla, con lo que eliminamos los puntos que no deban ser vistos.

Al igual que en el programa de dos dimensiones, se ajustarán las escalas al tamaño de la pantalla.

También en este caso dibujamos la función seno, sólo que esta vez en tres dimensiones.

```
10 REM *********************************
20 REM **                             **
30 REM **   PROGRAMA PARA AMSTRAD     **
40 REM **                             **
50 REM **    GRAFICAS EN 3 - D        **
60 REM **                             **
70 REM *********************************
80 MODE 1
90 CLS:CLG
95 REM INCIALIZA LOS VALORES
100 X0=270:Y0=250
110 XMIN=-4:XMAX=4
120 YMIN=-4:XMAX=4
130 ZMIN=-1:ZMAX=1
135 REM CALCULA  LA ESCALA
140 XSTEP=(XMAX-XMIN)/300:XSCALL=1/XSTEP
150 YSTEP=(YMAX-YMIN)/300:YSCALL=1/YSTEP
160 ZSCALL =200/(ZMAX-ZMIN)
170 FOR Y= YMIN TO YMAX STEP YSTEP
180 FOR X= XMIN TO XMAX STEP XSTEP
190 GOSUB 400
200 REM AJUSTAMOS LA ESCALA
210 X1=INT((X-XMIN)*XSCALL)
220 Y1=INT((Y-YMIN)*YSCALL)
230 Z1=INT((Z-ZMIN)*ZSCALL)
240 REM PROYECTAMOS
250 XP=X1-Z1*0.3+X0
260 YP=Y1-Z1*0.5+Y0
270 REM BORRA LA LINEA OCULTAS Y PINTA
280 MOVE  XP,YP
290 DRAW XP,400,0
300 PLOT XP,YP,1
310 NEXT X
```

```
320 NEXT Y
330 END
400 REM CALCULO DE LA FUNCION
410 Z=SIN(SQR(X*X+Y*Y))
420 RETURN
```

Programa 4.2

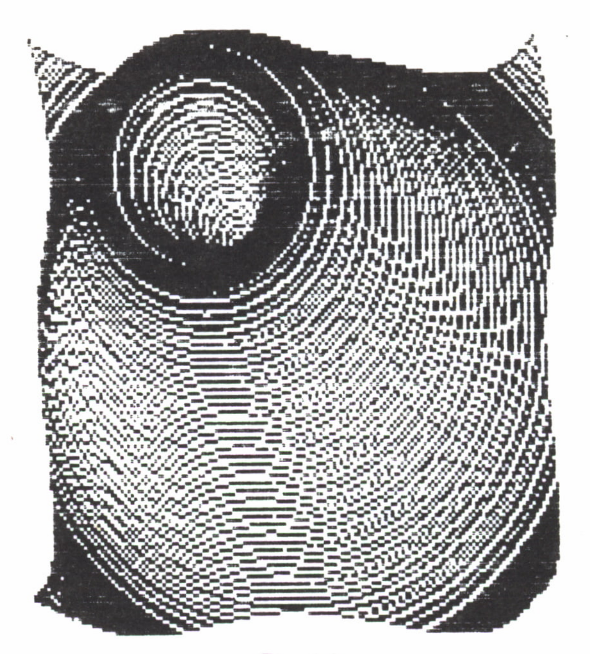


Fig. 4.2

Se observará que el tiempo que tarda el ordenador en completar la gráfica es bastante largo.

Para que éste sea ligeramente menor existe una forma que ahorra algo de tiempo, y nos da una idea clara de la gráfica.

Consiste en no representar todos los puntos, sino sólo algunos, como si se tomaran los puntos situados sobre una rejilla superpuesta a la gráfica.

Para ello la variable XS de la sentencia

FOR X = XMIN TO XMAX STEP XS

no será constante a lo largo de todo el programa, como en el caso anterior, sino que será variable a lo largo del programa.

Al igual que en los anteriores programas representaremos la función seno.

```
10 REM ****************************************
20 REM **                                    **
30 REM **      VERSION PARA IBM              **
40 REM **                                    **
50 REM **      GRAFICAS EN 3 - D             **
60 REM **                                    **
70 REM ****************************************
80 SCREEN 1
90 CLS
95 REM DEFINE LOS PARAMETROS
100 X0=100:Y0=50
110 I=5
120 XMIN=-4:XMAX=4
130 YMIN=-4:YMAX=4
140 ZMIN=-1:ZMAX=1
150 XSTEP=(XMAX - XMIN)/150:XSCALL = 1/XSTEP
160 YSTEP=(YMAX - YMIN)/150:YSCALL = 1/YSTEP
170 ZSCALL = 100 /(ZMAX - ZMIN)
180 FOR Y = YMIN TO YMAX STEP YSTEP
185 REM AJUSTA EL STEP PARA TARDAR MENOS
190 IF I<5 THEN I=I+1:XST =XSTEP * 5
200 IF I=5 THEN I=0:XST = XSTEP
210 FOR X = XMIN TO XMAX STEP XSTEP
220 GOSUB 330
225 REM AJUSTAMOS LA ESCALA
230 X1 = X0 + INT ((X-XMIN)*XSCALL)
240 Y1 = Y0 + INT ((Y-YMIN)*YSCALL)
250 Z1 = Z0 + INT ((Z-ZMIN)*ZSCALL)
260 REM PROYECTAMOS
270 XP = X1 - Z1 * .3
280 YP = Y1 - Z1 * .5
285 REM BORRA LAS LINEAS OCULTAS Y PINTA
290 LINE (XP,YP)-(XP,199),0
300 PSET (XP,YP)
310 NEXT X
320 NEXT Y
325 END
330 REM CALCULO DE LA FUNCION
340 Z = SIN (SQR (X*X+Y*Y))
350 RETURN
```

VARIACIONES PARA M.S.X.

```
LINEA  80 SCREEN 2
LINEA 120 XI=-4:XA=4
LINEA 130 YI=-4:YA=4
LINEA 140 ZI=-1:ZA=1
LINEA 150 XP=(XA-XI)/150:XL=1/XP
```

```
LINEA 160 YP=(YA-YI)/150:YL=1/YP
LINEA 170 ZL=100/(ZA-ZI)
LINEA 180 FOR Y=YI TO YA STEP YP
LINEA 190 IF I<5 THEN I=I+1:XT=XP*5
LINEA 200 IF I=5 THEN I=0:XT=XP
LINEA 210 FOR X=XI TO XA STEP XP
LINEA 230 X1=XO+INT((X-XI)*XL)
LINEA 240 Y1=YO+INT((Y-YI)*YL)
LINEA 250 Z1=ZO+INT((Z-ZI)*ZL)
```

Programa 4.3

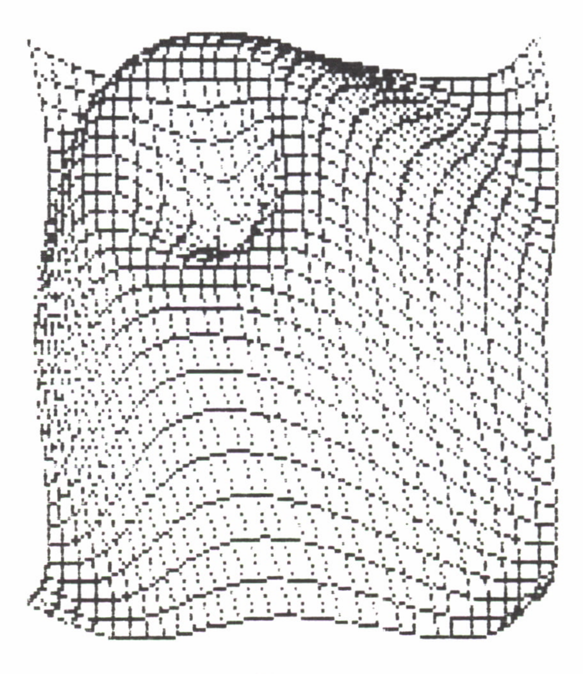


Fig. 4.3

En principio parece posible ampliar la representación de datos a un mayor número, pero esto implicaría ver en cuatro dimensiones lo cual, desgraciadamente, nos es imposible a los seres humanos por muchas proyecciones que hagamos.



HISTOGRAMAS

Cuando los datos a representar no necesitan mucha precisión, o cuando disponemos de pocos de ellos, es más práctico utilizar diagramas de barras o histogramas. Esto consiste en representar puntos «mas gordos» en forma de barras.
Su aplicación está muy extendida en el ámbito comercial para observar variaciones y tendencias de ventas, stocks, etc.

Los histogramas en dos dimensiones pueden hacerse con los caracteres, pero, debido a su poca vistosidad, lo realizamos como gráfica.

Veamos el ejemplo de tener ciertos datos en función del tiempo.

```
 10 REM ***************************
 20 REM **                       **
 30 REM **PROGRAMA PARA SPECTRUM **
 40 REM **                       **
 50 REM ** HISTOGRAMAS 2 - D     **
 60 REM **                       **
 70 REM ***************************
 80 REM INICIALIZAR VARIABLES
 90 READ N
100 DIM T(N)
110 FOR I=1 TO N
120 READ T(I)
130 NEXT I
140 FOR I=1  TO N
150 LET X=I*10+50
160 LET Y=T(I)*10+20
170 FOR J=1 TO 7
180 PLOT X,Y
190 DRAW 0,-T(I)*10
200 NEXT J
210 NEXT I
500 DATA 20,1,6,8,1,5,2,7,3,5,9,1,7,3,8,9,2,5,6,4,1
```

Programa 4.4

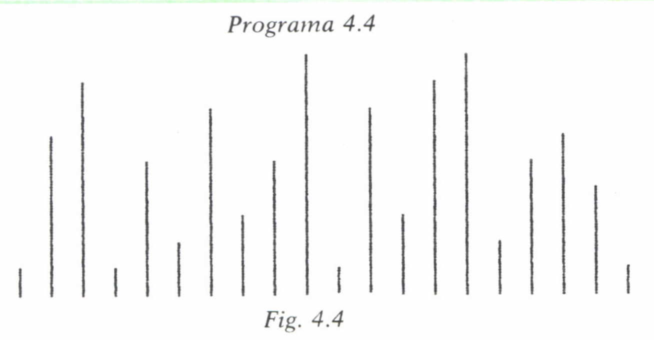


Fig. 4.4

Aunque no lo describiremos aquí, se pueden representar varios datos sobre un mismo gráfico utilizando barras de diferentes colores.

También existe la posibilidad de representar, en vez de barras, un dibujo del objeto sobre cuyos datos se estén representando; por ejemplo, espigas de distinta altura al representar la producción de trigo. Debido a lo poco que añade no realizaremos dicho programa.

La generalización a tres dimensiones es equivalente al caso de las gráficas en tres dimensiones, sustituyendo las barras por ortoedros.

Al igual que en el caso anterior, se proyectarán sobre un plano las coordenadas obtenidas.

```
10 REM ***************************************
20 REM **                                   **
30 REM **      VERSION PARA IBM             **
40 REM **                                   **
50 REM **      HISTOGRAMAS 3 - D            **
60 REM **                                   **
70 REM ***************************************
80 SCREEN 1
90 CLS
100 REM DEFINE LOS PARAMETROS
110 C=1
120 XO=100:YO=170
130 XMIN=-4:XMAX=4
140 ZMIN=-4:ZMAX=4
150 YMIN=-1:YMAX=1
160 XSCALL=150/(XMAX - XMIN) :XSTEP = 15 /XSCALL
170 ZSCALL=150/(ZMAX - ZMIN) :ZSTEP = 15 /ZSCALL
180 YSCALL = 100 /(YMAX - YMIN)
190 FOR Z = ZMAX TO ZMIN STEP -ZSTEP
200 FOR X = XMAX TO XMIN STEP -XSTEP
210 GOSUB 300
220 REM AJUSTAMOS LA ESCALA
230 XD = INT ((X-XMIN)*XSCALL)
240 YD = INT ((Y-YMIN)*YSCALL)
250 ZD = INT ((Z-ZMIN)*ZSCALL)
260 GOSUB 420
270 NEXT X
280 NEXT Z
290 END
300 REM CALCULO DE LA FUNCION
310 Y = SIN (SQR (X*X+Z*Z))
320 RETURN
330 REM RUTINA QUE UNE DOS PUNTOS
340 XP = X1 - Z1 * .5 + XO
350 YP = YO -Y1 - Z1 * .5
360 XP1= X2 - Z2 * .5  + XO
```

```
370 YP1= YO - Y2 - Z2 * .5
380 LINE (XP,YP)-(XP1,YP1),C
390 IF C=0 AND P=1 THEN LINE (XP,YP)-(XP1,YP1)
400 IF C=2 THEN LINE (XP,YP)-(XP1,YP1)
410 RETURN
420 REM RUTINA QUE DIBUJA CUBOS
430 C=2
440 X1=XD:Y1=YD:Z1=ZD
450 X2=XD:Y2=YD:Z2=ZD-7*ZSCALL/10
460 IX=XSCALL/10:IY=0:IZ=0
470 GOSUB 580
480 X1=XD:Y1=YD:Z1=ZD
490 X2=XD:Y2= 0:Z2=ZD
500 IX=0:IY=0:IZ=-ZSCALL/10:C=0
510 GOSUB 580
520 C=1
530 X1=XD:Y1=YD:Z1=ZD-ZSCALL*7/10
540 X2=XD:Y2= 0:Z2=ZD-ZSCALL*7/10
550 IX=XSCALL/10:IY=0:IZ=0
560 GOSUB 580
570 RETURN
580 REM RUTINA DE HACER CARAS
590 FOR P=1 TO 7
600 X1=X1+IX:X2=X2+IX
610 Y1=Y1+IY:Y2=Y2+IY
620 Z1=Z1+IZ:Z2=Z2+IZ
630 GOSUB 330
640 NEXT P
650 RETURN
```

VARIACIONES PARA M.S.X.

```
LINEA  80 SCREEN 2
LINEA 130 XI=-4:XA=4
LINEA 140 ZI=-4:ZA=4
LINEA 150 YI=-1:YA=1
LINEA 160 XL=150/(XA-XI):XP=15/XL
LINEA 170 ZL=150/(ZA-ZI):ZP=15/ZL
LINEA 180 YL=100/(YA-YI)
LINEA 190 FOR Z=ZA TO ZI STEP -ZP
LINEA 200 FOR X=XA TO XI STEP -XP
LINEA 230 XD=INT((X-XI)*XL)
LINEA 240 YD=INT((Y-YI)*YL)
LINEA 250 ZD=INT((Z-ZI)*ZL)
LINEA 260 XQ=X2-Z2*0.5+XO
LINEA 370 YQ=YO-Y2-Z2*0.5
LINEA 380 LINE(XP,YP)-(XQ,YQ),C
```

```
LINEA 390 IF C=0 AND P=1 THEN LINE(XP,YP)-(XQ,
YQ)
LINEA 400 IF C=2 THEN LINE(XP,YP)-(XQ,YQ)
LINEA 450 X2=XD:Y2=YD:Z2=ZD-7*ZL/10
LINEA 460 IX=XL/10:IY=0:IZ=0
LINEA 500 IX=0:IY=0:IZ=-ZL/10:C=0
LINEA 530 X1=XD:Y1=YD:Z1=ZD-ZL*7/10
LINEA 540 X2=XD:Y2=0 :Z2=ZD-ZL*7/10
LINEA 550 IX=XL/10:IY=0:IZ=0
```

Programa 4.5

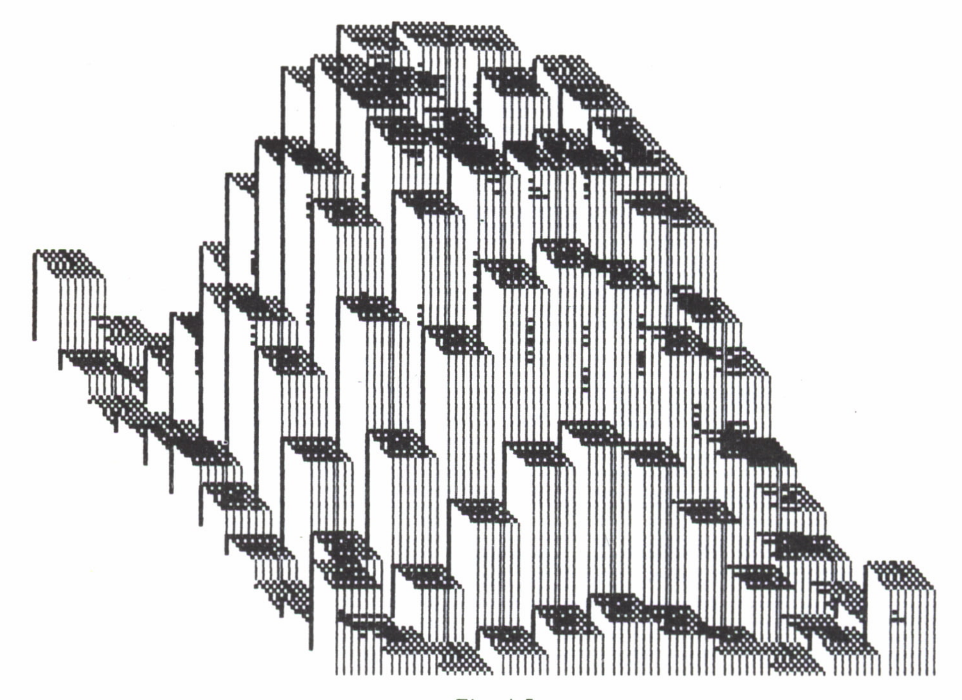


Fig. 4.5

Al igual que antes, en vez de ortoedros se pueden trazar otro tipo de figuras, tales como barras, líneas, etc.; pero tampoco lo realizaremos, por lo poco que añadiría, ya que consistiría en variar el dibujo del ortoedro por la figura que deseemos.

TRANSFORMACIONES Y MATRICES 5



INTRODUCCION

CON todo lo que hemos visto hasta ahora, sobre todo en el tercer capítulo, somos ya capaces de visualizar en la pantalla del ordenador desde cualquier punto de vista, pero no sabemos nada todavía de cómo mover un objeto en el espacio para luego pintarlo. En este capítulo veremos cómo se hace esto, además de establecer una notación más homogénea con la que realizar los cálculos sobre los puntos del objeto a representar. Aprenderemos a trasladar un objeto en el espacio, girarlo sin cambiar el punto de vista, aumentar o disminuir su tamaño, y a deformarlo para, a partir de un objeto simple, obtener figuras más complicadas.

Antes de entrar en detalles sobre cada una de estas operaciones vamos a explicar el concepto general de transformación. Una transformación es un conjunto de operaciones que convierten algo en otra cosa distinta. En nuestro caso vamos a convertir un punto del espacio en otro punto del espacio con coordenadas diferentes. Al conjunto de operaciones y constantes necesarias para realizar esto se le llama *transformación*.



ESCALADO

La transformación más sencilla que podemos hacer es aumentar o disminuir el tamaño del objeto de forma uniforme con un factor de escala dado.

Para conseguir esto lo único que tenemos que hacer es multiplicar cada una de las coordenadas del punto por un factor de escala constante. Si este razonamiento lo aplicamos a cada punto del objeto tendremos un nuevo objeto más grande o más pequeño que el original. Para conseguir un cuerpo más grande el factor de escala tendrá que ser mayor que 1, y para dis-

minuir el tamaño del objeto, el factor deberá ser menor que 1; por ejemplo, si el factor de escala vale 0,5 conseguiremos un cuerpo de la mitad de tamaño que el original.

Las fórmulas para conseguir este efecto son:

$$XT = S*X$$
$$YT = S*Y$$
$$ZT = S*Z$$

Donde S es el factor de escala y X,Y,Z las coordenadas del punto original y XT, YT, ZT las coordenadas del punto transformado.

Si el factor de escala lo hacemos negativo conseguiremos una «reflexión» del objeto, cambiaremos la parte de arriba con la de abajo y la parte izquierda con la derecha, como si viéramos su imagen en un espejo.

El factor de escala no tiene por qué ser igual para todas las coordenadas; X,Y,Z pueden tener factores de escala distinta, con lo que conseguiremos cambiar el tamaño del objeto de una forma distinta con respecto a cada dimensión, o sea, lo deformaremos. Esto puede ser útil para, por ejemplo, a partir de un cubo, obtener cualquier tipo de paralelepípedo.

Una aplicación de esta transformación la podemos ver en el siguiente programa:

```
10 REM ****************************************
20 REM **                                    **
30 REM **        VERSION PARA IBM            **
40 REM **                                    **
50 REM **        CAMBIO DE ESCALA            **
60 REM **                                    **
70 REM ****************************************
80 SCREEN 1
90 CLS
95 PI=3.141592654#
100 XO=160:YO=100
105 D=200
110 A=30*PI/180:B=30*PI/180
120 A1=COS(A):A2=SIN(A)
130 B1=COS(B):B2=SIN(B)
140 LOCATE 25,1
150 INPUT "ESCALA (X,Y,Z) ";SX,SY,SZ
160 CLS
170 FOR I=1 TO 2
180 IF I=2 THEN RESTORE
190 REM VER SI HA ACABADO
200 READ COLORP
210 IF COLORP=-1 THEN GOTO 385
220 REM LEER EL PUNTO
```

```
230 READ X,Y,Z
240 IF I=2 THEN X=X*SX:Y=Y*SY:Z=Z*SZ
250 REM CALCULAR LA PROYECCION
260 X=A1*X-A2*Z
270 Y=Y*B1-Z*A1*B2-X*A2*B2
280 Z=Z*A1*B1+X*A2*B1+Y*B2
290 PX =D*X/(D-Z)
300 PY =D*Y/(D-Z)
310 REM TRASLADAR EL ORIGEN
320 X1=PX+X0
330 Y1=PY+Y0
340 REM VER SI HAY QUE DIBUJAR
350 IF COLORP= 0 THEN PRESET (X1,Y1)
360 IF COLORP= 1 THEN LINE - (X1,Y1)
370 REM REPETIR EL PROCESO
380 GOTO 200
385 NEXT I
390 DATA 0,0,0,0,1,50,0,0,1,50,0,50,1,0,0,50
400 DATA 1,0,0,0,1,25,50,25,1,0,0,50,0,50,0,50
410 DATA 1,25,50,25,1,50,0,0,1,25,-50,25,1,0,0,0
420 DATA 1,25,-50,25,0,0,0,50,1,25,-50,25
430 DATA 1,50,0,50,-1
440 GOTO 440

VARIACIONES PARA M.S.X.

LINEA  80 CLS

LINEA  90 LOCATE 20,1:INPUT "ESCALA (X,Y,Z)
";SX,SY,SZ
LINEA 140 SCREEN 2

BORRAR LA LINEA 150
```

Programa 5.1

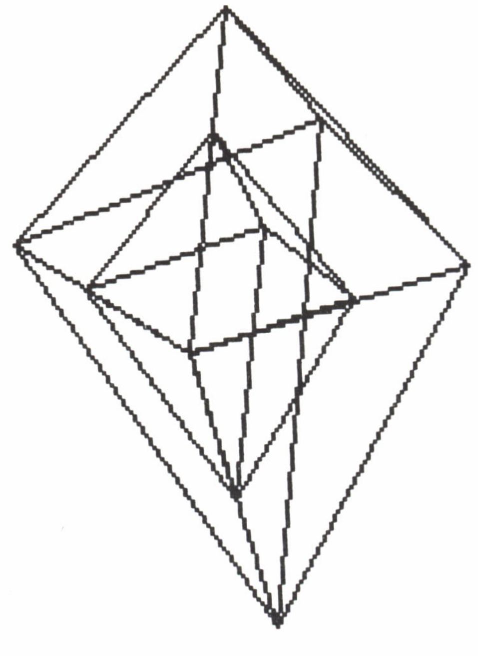


Fig. 5.1

Este programa pinta primero el objeto sin modificar, y luego pide tres valores que van a ser los tres factores de escala con los que transformar la figura, pintando la figura transformada encima.



ROTACION

Otra transformación muy usada consiste en girar el cuerpo un determinado ángulo alrededor de uno de los tres ejes de coordenadas, con lo que conseguimos ver el objeto en distintas posiciones sin necesidad de cambiar el punto de vista.

Para girar un ángulo alrededor del eje Z, por ejemplo, aplicamos las siguientes fórmulas:

XT = X*COS(a) + Y*SIN(a)
YT = -X*SIN(a) + Y*COS(a)

Con estas fórmulas conseguimos girar el objeto alrededor del eje Z en el sentido horario. Como podemos observar, la coordenada Z no sufre ningún cambio, ya que al girar sobre este eje permanece constante su valor.

Para girar sobre otros ejes las fórmulas son:

= GIRO ALREDEDOR DEL EJE Y:
XT = X*COS(a) + Z*SIN(a)
ZT = -X*SIN(a) + Z*COS(a)
= GIRO ALREDEDOR DEL EJE X:
YT = Y*COS(a) + Z*SIN(a)
ZT = -Y*SIN(a) + Z*COS(a)

En el siguiente programa podremos comprobar el funcionamiento de esta transformación:

```
10 REM ******************************
20 REM **                          **
30 REM **   PROGRAMA PARA AMSTRAD  **
40 REM **                          **
50 REM **      ROTACION            **
60 REM **                          **
70 REM ******************************
80 MODE 1
90 CLS:CLG
100 X0=320:Y0=200:d=500
110 a=10:b=50
115 A1=COS(a):A2=SIN(a)
117 B1=COS(b):b2=SIN(b)
120 LOCATE 1,25
130 INPUT "ANGULO ->";C
131 C=C*PI/180:C1=COS(C):C2=SIN(C)
140 FOR i=1 TO 2
150 IF i=2 THEN RESTORE
160 REM LEER SI SE ACABA
170 READ COLOR
180 IF COLOR = -1 THEN GOTO 355
190 REM LEER EL PUNTO
200 READ X,Y,Z
210 IF i=2 THEN X1=X*C1+Z*C2:Z=Z*C1-X*C2:X=X1
220 REM CALCULAR LA PROYECCION
230 X1=A1*X-A2*Z
240 Y1=Y*B1-Z*A1*B2-X*A2*B2
250 Z1=Z*A1*B1+X*A2*B1+Y*B2
260 X1=D*X1/(D-Z1)
270 Y1=D*Y1/(D-Z1)
280 REM TRASLADAR EL ORIGEN
290 X1=X1+X0
300 Y1=Y1+Y0
310 REM VER SI HAY QUE DIBUJAR
320 IF COLOR = 0 THEN MOVE X1,Y1
330 IF COLOR = 1 THEN DRAW X1,Y1
```

```
340 REM REPETIR EL PROCESO
350 GOTO 170
355 NEXT i
360 DATA 0,0,0,100 ,1,0,10,25,1,300,0,0,0,0,10,25
370 DATA 1,0,40,0,1,0,83,0,1,33,83,0,1,65,35,0
380 DATA 0,0,40,0,1,300,0,0,0,150,4,0,1,30,0,-100
385 DATA 1,0,0,-100,1,0,10,-25,1,300,0,0,0,0,10,-25
390 DATA 1,0,40,0
400 DATA 0,0,0,-100,1,0,0,100,1,30,0,100,1,150,4,0,-1
410 GOTO 410
```

Programa 5.2

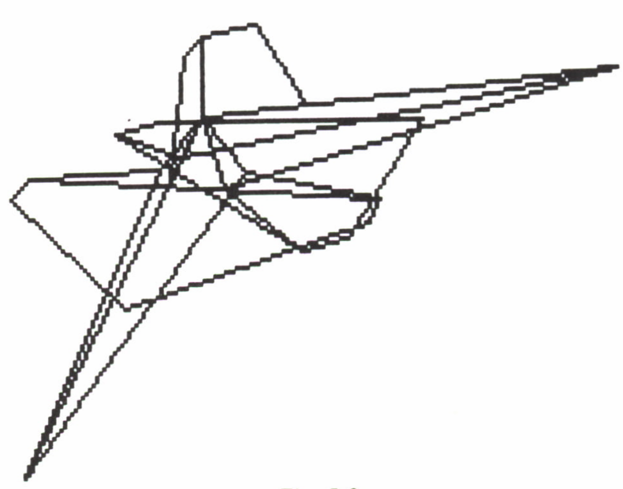


Fig. 5.2

Al ponerlo en marcha nos pedirá un ángulo con el que rotar la figura alrededor del eje Y y la pintará rotada. Cambiando las fórmulas de rotación podemos girar el cuerpo sobre cualquiera de los tres ejes.



TRASLACION

Trasladar un objeto de una posición a otra consiste simplemente en sumar o restar un vector de coordenadas XM,YM,ZM a todos los puntos de la figura, con lo que las fórmulas quedan en este caso:

$$XT = X + XM$$
$$YT = Y + YM$$
$$ZT = Z + ZM$$

En estas fórmulas tan sencillas podemos poner el cuerpo en el lugar del espacio que queramos.

El programa descrito a continuación hace precisamente eso, trasladar a un punto del espacio nuestro objeto:

```
10 REM ****************************************
20 REM **                                    **
30 REM **       VERSION PARA IBM             **
40 REM **                                    **
50 REM **          TRASLADAR                 **
60 REM **                                    **
70 REM ****************************************
80 SCREEN 1
90 CLS
100 PI=3.141592654#
110 X0=160:Y0=100
120 DIS=300
130 A=-30*PI/180:B=30*PI/180
140 A1=COS(A):A2=SIN(A)
150 B1=COS(B):B2=SIN(B)
160 LOCATE 25,1
170 INPUT "DESPLAZAMIENTOS (X,Y,Z)";XM,YM,ZM
180 CLS
190 FOR I=1 TO 2
200 IF I=2 THEN RESTORE
210 REM VER SI HA ACABADO
220 READ COLORP
230 IF COLORP=-1 THEN GOTO 440
240 REM LEER EL PUNTO
250 READ X,Y,Z
260 X=X*.3:Y=Y*.3:Z=Z*.3
270 IF I=1 THEN   300
280 REM TRASLADA
290 X=X+XM:Y=Y+YM:Z=Z+ZM
300 REM CALCULAR LA PROYECCION
310 X=A1*X-A2*Z
320 Y=Y*B1-Z*A1*B2-X*A2*B2
330 Z=Z*A1*B1+X*A2*B1+Y*B2
340 PX =DIS*X/(DIS-Z)
350 PY =DIS*Y/(DIS-Z)
360 REM TRASLADAR EL ORIGEN
370 X1=PX+X0
380 Y1=Y0-PY
390 REM VER SI HAY QUE DIBUJAR
400 IF COLORP= 0 THEN PSET (X1,Y1)
410 IF COLORP= 1 THEN LINE - (X1,Y1)
420 REM REPETIR EL PROCESO
430 GOTO 220
```

```
440 NEXT I
450 DATA 0,0,0,100,1,0,10,25,1,300,0,0,0,0,10,25
460 DATA 1,0,40,0,1,0,83,0,1,33,83,0,1,65,35,0
470 DATA 0,0,40,0,1,300,0,0,0,150,4,0,1,30,0,-100
480 DATA 1,0,0,-100,1,0,10,-25,1,300,0,0,0,0,10,-25
490 DATA 1,0,40,0
500 DATA 0,0,0,-100,1,0,0,100,1,30,0,100,1,150,4,0,-1
510 GOTO 510

VARIACIONES PARA M.S.X.

LINEA  80 CLS
LINEA  90 LOCATE 20,1:INPUT "DESPLAZAMIENTOS
(X,Y,Z) ";XM,YM,ZM
LINEA 160 SCREEN 2

BORRAR LA LINEA 170
```

Programa 5.3

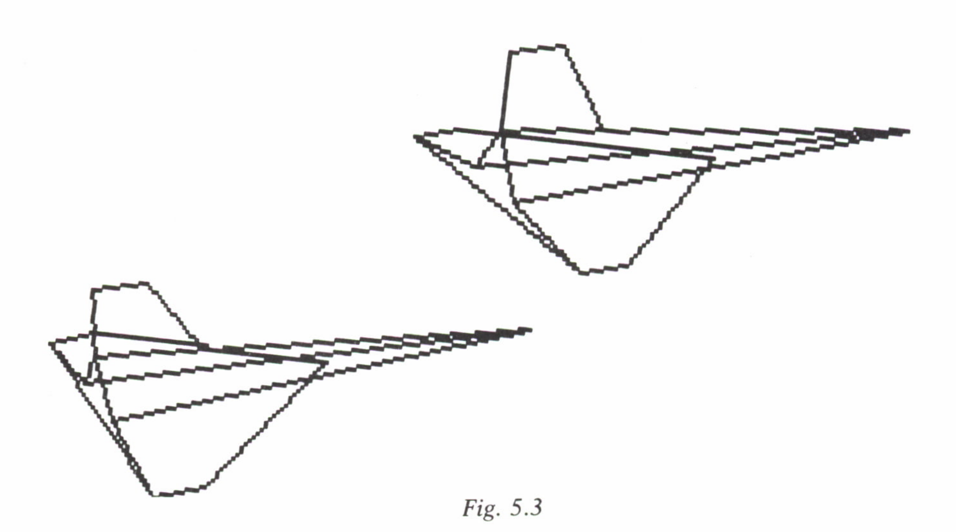


Fig. 5.3

Hay que tener cuidado en no dar unas coordenadas muy grandes al punto en el cual queremos trasladar nuestra figura, ya que podría llegar a salirse de los límites de la pantalla.

RESUMEN DE LAS TRANSFORMACIONES

ESCALADO:

XT = X*S1
YT = Y*S2
ZT = Y*S3

ROTACION:

— SOBRE EL EJE Z;
XT = X*COS(a) + Y*SIN(a)
YT = -X*SIN(a) + Y*COS(a)
— SOBRE EL EJE Y;
XT = X*COS(a) + Z*SIN(a)
ZT = -X*SIN(a) + Z*COS(a)
— SOBRE EL EJE X;
YT = Y*COS(a) + Z*SIN(a)
ZT = -Y*SIN(a) + Z*COS(a)

TRASLACION:

XT = X + XM
YT = Y + YM
ZT = Z + ZM

La fórmula general para una transformación de cualquier tipo sería:

XT = AX*X + BX*Y + CX*Z + KX
YT = AY*X + BY*Y + CY*Z + KY
ZT = AZ*X + BZ*Y + CZ*Z + KZ

Donde AX, AY, AZ, BX, BY, BZ, CX, CY, CZ, KX, KY, KZ serían cualquier tipo de números o expresiones matemáticas.

Esta fórmula general nos dice que la coordenada transformada es función de las tres coordenadas del punto más una constante o fórmula.



COMBINACION DE TRANSFORMACIONES

Por ahora hemos hecho transformaciones simples, pero nada impide que realicemos cambios más complejos combinando diversas transformaciones.

El proceso es muy sencillo; sólo hay que ir aplicando en cadena cada una de las fórmulas que conocemos para realizar una transformación compleja. Unicamente hay que tener cuidado con una cosa, y es que la trasla-

ción hay que realizarla en último lugar, realizando antes giros y escaladas, ya que si no los resultados no serían normalmente los deseados.

Los pasos a seguir habitualmente serían los siguientes:

— Traslación del objeto al centro de la pantalla si no estuviera definido en el origen. Esto se realiza aplicando una traslación con el vector -XM, -YM, -ZM, donde XM, YM, ZM es la posición del CENTRO GEOMETRICO del cuerpo.

— Una vez está el objeto en el origen de coordenadas podemos rotarlo o escalarlo como queramos, teniendo en cuenta que en cada transformación hay que utilizar como coordenadas de partida los valores producidos por la transformación anterior.

— Trasladarlo al punto que deseemos, el mismo que ocupaba u otro distinto donde queramos colocarlo.

Por ejemplo, en el programa siguiente cogemos una figura e introduciendo unos datos que nos irá pidiendo lo giramos alrededor de los tres ejes, lo escalamos una cierta cantidad y, por último, lo trasladamos a una determinada posición del espacio.

```
10 REM ****************************************
20 REM **                                    **
30 REM **      VERSION PARA IBM              **
40 REM **                                    **
50 REM **      AHORA TODO JUNTO              **
60 REM **                                    **
70 REM ****************************************
80 SCREEN 1
90 CLS
100 PI=3.141592654#
110 XO=160:YO=100
120 DIS=300
130 A=-30*PI/180:B=30*PI/180
140 A1=COS(A):A2=SIN(A)
150 B1=COS(B):B2=SIN(B)
160 LOCATE 25,1
170 INPUT "GIROS A,B,C";C,D,E
180 C=C*PI/180:D=D*PI/180:E=E*PI/180
190 C1=COS(C):C2=SIN(C)
200 D1=COS(D):D2=SIN(D)
210 E1=COS(E):E2=SIN(E)
220 LOCATE 25,1
230 INPUT "DESPLAZAMIENTOS (X,Y,Z)";XM,YM,ZM
240 CLS
250 LOCATE 25,1
260 INPUT "ESCALA (X,Y,Z) ";SX,SY,SZ
```

```
270 CLS
280 FOR I=1 TO 2
290 IF I=2 THEN RESTORE
300 REM VER SI HA ACABADO
310 READ COLORP
320 IF COLORP=-1 THEN GOTO 580
330 REM LEER EL PUNTO
340 READ X,Y,Z
345 X=X*.5:Y=Y*.5:Z=Z*.5
350 IF I=1 THEN   440
360 REM GIRA
370 Y1=Y*C1+Z*C2:Z=Z*C1-Y*C2:Y=Y1
380 X1=X*D1+Z*D2:Z=Z*D1-X*D2:X=X1
390 X1=X*E1+Y*E2:Y=Y*E1-X*E2:X=X1
400 REM CAMBIA LA ESCALA
410 X=X*SX:Y=Y*SY:Z=Z*SZ
420 REM TRASLADA
430 X=X+XM:Y=Y+YM:Z=Z+ZM
440 REM CALCULAR LA PROYECCION
450 X=A1*X-A2*Z
460 Y=Y*B1-Z*A1*B2-X*A2*B2
470 Z=Z*A1*B1+X*A2*B1+Y*B2
480 PX =DIS*X/(DIS-Z)
490 PY =DIS*Y/(DIS-Z)
500 REM TRASLADAR EL ORIGEN
510 X1=PX+X0
520 Y1=Y0-PY
530 REM VER SI HAY QUE DIBUJAR
540 IF COLORP= 0 THEN PSET (X1,Y1)
550 IF COLORP= 1 THEN LINE - (X1,Y1)
560 REM REPETIR EL PROCESO
570 GOTO 310
580 NEXT I
590 DATA 0,0,0,100,1,0,10,25,1,300,0,0,0,0,10,25
600 DATA 1,0,40,0,1,0,83,0,1,33,83,0,1,65,35,0
610 DATA 0,0,40,0,1,300,0,0,0,150,4,0,1,30,0,-100
620 DATA 1,0,0,-100,1,0,10,-25,1,300,0,0,0,0,10,-25
630 DATA 1,0,40,0
640 DATA 0,0,0,-100,1,0,0,100,1,30,0,100,1,150,4,0,-1
650 GOTO 650
```

VARIACIONES PARA M.S.X.

```
LINEA  80 CLS
LINEA  90 LOCATE 20,1:INPUT "GIOROS A,B,C";C,D,E
LINEA 160 CLS
LINEA 170 LOCATE 20,1:INPUT "DESPLAZAMIENTOS
(X,Y,Z) ";XM,YM,ZM
LINEA 220 CLS
```

```
LINEA 230 LOCATE 20,1:INPUT "ESCALA (X,Y,Z)
";SX,SY,SZ
LINEA 240 SCREEN 2

QUITAR LÀS LINEAS 250 Y 260
```

Programa 5.4

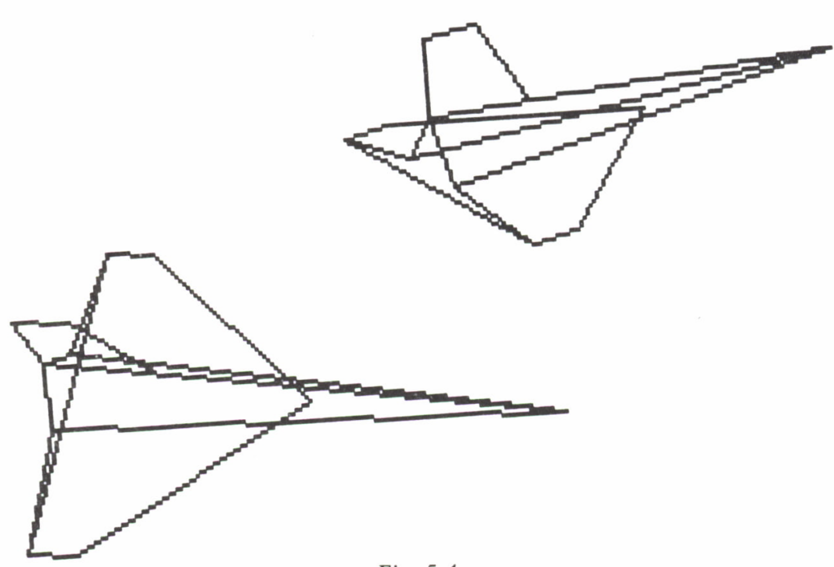


Fig. 5.4

El esquema general de cualquier programa sería el siguiente:

```
OPERACIONES PRELIMINARES (BORRADO DE PANTALLA, ETC.)
INTRODUCCION DE DATOS
INICIALIZACION DE VARIABLES Y CONSTANTES
REPITE HASTA QUE COLOR = -1
    LEE UN PUNTO
                                              | GIRO
    APLICA DIVERSAS TRANSFORMACIONES          | TRASLACION
                                              | ESCALADO
    CALCULA LAS PROYECCIONES (PX,PY)
    TRASLADA AL CENTRO DE LA PANTALLA
    PINTA O MUEVE A ESE PUNTO
FIN DEL REPITE
FIN DEL PROGRAMA
```

Este esquema general que damos es muy parecido al que vimos en el capítulo tercero, añadiendo sólo la parte de transformaciones al objeto que tengamos que pintar.



CALCULO CON MATRICES

Ya conocemos las fórmulas que nos permiten hacer transformaciones sencillas y combinaciones de ellas, pero las hemos aplicado de una forma que, aunque simple, es un poco tediosa, porque debemos aplicar una detrás de otra a los puntos que queremos transformar. Para evitar esto podríamos combinar en una sola fórmula todas las transformaciones que vayamos a aplicar.

Para ello vamos a utilizar la notación matricial. Una matriz consiste en un conjunto de números ordenados según una tabla de filas y columnas, igual que las variables con subíndice del BASIC. Un ejemplo de matriz sería:

$$\begin{Vmatrix} 1 & 11 & -3{,}4 \\ 2 & 8 & -2 \\ 3 & 9 & 2{,}8 \\ 5 & 6 & 0 \end{Vmatrix}$$

El concepto es sencillo de entender, pero lo que ya no es tan sencillo es la forma de operar con ellas. El lector que no tenga conocimientos suficientes de matemáticas puede pasar por alto esta sección sin perder ninguna de las posibilidades, o bien puede leer el apéndice donde se especifican las operaciones con matrices y algunos conceptos útiles sobre ellas.

Según este tipo de notación, aplicar una transformación a un punto en el espacio sería multiplicar el trío de valores que son sus coordenadas por una matriz de transformación.

$$(XT,YT,ZT) = (X,Y,Z) \cdot \begin{Vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{Vmatrix}$$

donde las letras a...i son los coeficientes de la transformación.

Según esta fórmula general, las fórmulas para la transformación de escalado quedarían:

$$(XT,YT,ZT) = (X,Y,Z) \cdot \begin{Vmatrix} S1 & 0 & 0 \\ 0 & S2 & 0 \\ 0 & 0 & S3 \end{Vmatrix}$$

Esta fórmula es equivalente a las tres dadas en el apartado de escalado.

Igualmente para una rotación sobre el eje Z sería:

$$(XT,YT,ZT) = (X,Y,Z) \cdot \begin{Vmatrix} COS(a) & SIN(a) & 0 \\ -SIN(a) & COS(a) & 0 \\ 0 & 0 & 1 \end{Vmatrix}$$

Pero esta notación no sirve para aplicar la fórmula de la traslación, porque los coeficientes para sumar al final de las ecuaciones generales de transformación no salen por ningún sitio. Para evitar esto se utiliza un sistema de coordenadas homogéneas. En este sistema, además de tener el trío de coordenadas (X,Y,Z), tenemos una cuarta coordenada que vamos a llamar T y que siempre tomará el valor 1. De esta forma, las coordenadas de un punto del espacio quedan de la siguiente manera (X,Y,Z,1) y las matrices de transformación, en vez de tener tres filas y tres columnas, tienen cuatro filas y cuatro columnas.

Con este nuevo sistema de coordenadas las fórmulas para las transformaciones simples quedan:

ESCALADO

$$(XT,YT,ZT,1) = (X,Y,Z,1) \cdot \begin{Vmatrix} S1 & 0 & 0 & 0 \\ 0 & S2 & 0 & 0 \\ 0 & 0 & S3 & 0 \\ 0 & 0 & 0 & 1 \end{Vmatrix}$$

ROTACION

— SOBRE EL EJE Z

$$(XT,YT,ZT,1) = (X,Y,Z,1) \cdot \begin{Vmatrix} COS(a) & SIN(a) & 0 & 0 \\ -SIN(a) & COS(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Vmatrix}$$

— SOBRE EL EJE Y

$$(XT,YT,ZT,1) = (X,Y,Z,1) * \begin{Vmatrix} COS(a) & 0 & -SIN(a) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -SIN(a) & COS(a) & 0 \\ 0 & 0 & 0 & 1 \end{Vmatrix}$$

— SOBRE EL EJE X

$$(XT,YT,ZT,1) = (X,Y,Z,1) * \begin{Vmatrix} 1 & 0 & 0 & 0 \\ 0 & COS(a) & SIN(a) & 0 \\ -SIN(a) & 0 & COS(a) & 0 \\ 0 & 0 & 0 & 1 \end{Vmatrix}$$

TRASLACION

$$(XT,YT,ZT,1) = (X,Y,Z,1) * \begin{Vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ XM & YM & ZM & 1 \end{Vmatrix}$$

Al desarrollar estas fórmulas nos quedan exactamente las mismas que las que planteamos al definir las transformaciones.

La ventaja del uso de matrices es que si queremos combinar varias transformaciones sólo tenemos que multiplicar entre sí las matrices correspondientes a las transformaciones simples; por ejemplo, para hacer una traslación del punto (3,3,3) al origen y rotar el resultado 45 grados en el eje Z, podemos hacer:

$$A = \begin{Vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & -3 & -3 & 1 \end{Vmatrix} * \begin{Vmatrix} COS(PI/4) & SIN(PI/4) & 0 & 0 \\ -SIN(PI/4) & COS(PI/4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Vmatrix}$$

Siendo A la matriz resultado del producto, y ahora basta con aplicar esta matriz, que representa una transformación compleja al vector.

$$(XT,YT,ZT,1) = (X,Y,Z,1) * A$$

Sólo debemos tener en cuenta una condición, que la multiplicación de matrices no es conmutativa, por lo que no es lo mismo girar y luego trasladar que primero trasladar y luego girar.

Por supuesto, se pueden encadenar más de dos transformaciones siempre que el resultado sea una sola matriz para aplicarla luego a los puntos del objeto.

Esta notación matricial nos permite incluso generalizar más el concepto de transformación, incluyendo en ella los cálculos necesarios para transformar los puntos del espacio, tal y como son en realidad, a puntos del espacio desde el punto de vista del observador, englobando en éstas cálculos entonces todos los giros y traslaciones de los que hablábamos en el último apartado del capítulo 3. Con lo que entonces podremos introducir la matriz de transformación, que llamaremos matriz de transformación del punto de vista en los cálculos generales de la transformación. Entonces sólo deberemos aplicar las fórmulas para calcular la perspectiva, que sí son independientes, obteniendo los puntos en dos dimensiones en la pantalla.

Para evitar dolores de cabeza y tiempo a los lectores, ponemos aquí la matriz correspondiente a esta última transformación:

$$\begin{Vmatrix} COS(a) & -SIN(a)SIN(b) & SIN(a)COS(b) & 0 \\ 0 & COS(b) & SIN(b) & 0 \\ -SIN(a) & -COS(a)SIN(b) & COS(a)COS(b) & 0 \\ 0 & 0 & 0 & 1 \end{Vmatrix}$$

donde R es la distancia del observador al origen; a, el ángulo que forma con el plano XZ; b, el ángulo que forma con el eje y.

De todas formas, hay que recordar que esto es sólo una notación y solamente eso; las fórmulas y cálculos son las mismas, y muchas veces es más fácil y legible escribir en un programa las cosas paso a paso aunque sea un poco más lento.

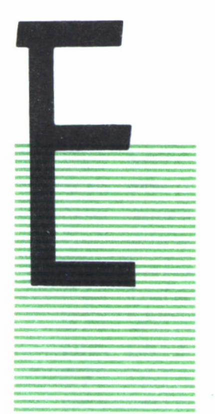
En algunos programas del capítulo 3 y de éste nos habremos dado cuenta de que al introducir unos datos determinados el programa no funciona, o si lo hace, lo hace mal. Esto es porque en ellos no hemos tenido en cuenta algunas cosas especiales como, por ejemplo, que la distancia del observador al plano de proyección sea cero, u otros similares. Todos estos problemas son fácilmente subsanables por el lector.

Un caso que puede darse es que el punto de observación en la proyección cónica se encuentre dentro del objeto que queremos representar, entonces los resultados son francamente desastrosos y habría que diseñar un sistema para evitar dibujar la parte del cuerpo que está detrás del observador.

APLICACIONES DIVERSAS 6

INTRODUCCION





EN este capítulo vamos a realizar una serie de ejemplos, como aplicación de las técnicas de representación, que ya conocemos. Aprenderemos, por ejemplo, a construir un cuerpo en tres dimensiones a partir de un dibujo en dos dimensiones, por rotación de esta figura sobre un eje o por traslación. También veremos cómo mover figuras para crear efectos animados en tres dimensiones, y cómo representar varios objetos a la vez en la pantalla. Estas son algunas de las aplicaciones típicas que podemos realizar con los gráficos de tres dimensiones, pero no son las únicas; nosotros podemos crear cualquier aplicación que se nos ocurra, desarrollando nuestros propios programas, aplicando las técnicas y procesos que hemos aprendido en los capítulos anteriores.



OBJETOS EN MOVIMIENTO

Crear la ilusión de movimiento en tres dimensiones en la pantalla de nuestro ordenador es una de las aplicaciones más vistosas que podemos llegar a realizar y, sin embargo, es también una de las más sencillas de programar.

El movimiento en el ordenador se puede simular pintando un objeto en una posición determinada, borrándolo y volviéndolo a pintar ligeramente desplazado de su posición inicial. Para conseguir esto vamos a utilizar una forma especial en la pantalla que emplea una operación lógica, el OR exclusivo, **XOR**; que si al pintar un punto en la pantalla éste no estaba pintado, lo pinta con su color correspondiente, y si ya estaba pintado, lo borra dejando el color del fondo. Gracias a esta opción la operación de borrado consiste simplemente en volver a pintar el objeto donde esta-

ba. En el Spectrum esta opción se consigue con la instrucción OVER 1, en Amstrad con PRINT CHR$(23) + CHR$(1) y en el caso de que nuestro ordenador no tuviera opción XOR, lo que haremos será pintar primero el objeto con el color adecuado para luego borrarlo pintándolo en el color del fondo.

Para mover el objeto aplicaremos cualquiera de las transformaciones que ya conocemos; por ejemplo, el siguiente programa mueve un objeto alrededor del eje Z:

```
10 REM ****************************************
20 REM **                                    **
30 REM **        VERSION PARA IBM            **
40 REM **                                    **
50 REM **        GIRANDO EL OBJETO           **
60 REM **                                    **
70 REM ****************************************
80 SCREEN 1
90 CLS
100 PI=3.141592654#
110 XO=160:YO=100
120 DIS=300
130 INE=-15*PI/180
140 A=-30*PI/180:B=30*PI/180
150 A1=COS(A):A2=SIN(A)
160 B1=COS(B):B2=SIN(B)
170 C=0:D=0:E=0:XM=-100:YM=-100:ZM=-50
180 SX=1:SY=1:SZ=1
190 C=C*PI/180:D=D*PI/180:E=E*PI/180
200 C1=COS(C):C2=SIN(C)
210 D1=COS(D):D2=SIN(D)
220 E1=COS(E):E2=SIN(E)
230 FOR I=1 TO 2
240 RESTORE
250 REM VER SI HA ACABADO
260 READ COLORP
270 IF COLORP=-1 THEN GOTO 530
280 REM LEER EL PUNTO
290 READ X,Y,Z
300 X=X*.5:Y=Y*.5:Z=Z*.5
310 REM GIRA
320 Y1=Y*C1+Z*C2:Z=Z*C1-Y*C2:Y=Y1
330 X1=X*D1+Z*D2:Z=Z*D1-X*D2:X=X1
340 X1=X*E1+Y*E2:Y=Y*E1-X*E2:X=X1
350 REM CAMBIA LA ESCALA
360 X=X*SX:Y=Y*SY:Z=Z*SZ
370 REM TRASLADA
```

```
380 X=X+XM:Y=Y+YM:Z=Z+ZM
390 REM CALCULAR LA PROYECCION
400 X=A1*X-A2*Z
410 Y=Y*B1-Z*A1*B2-X*A2*B2
420 Z=Z*A1*B1+X*A2*B1+Y*B2
430 PX =DIS*X/(DIS-Z)
440 PY =DIS*Y/(DIS-Z)
450 REM TRASLADAR EL ORIGEN
460 X1=PX+X0
470 Y1=Y0-PY
480 REM VER SI HAY QUE DIBUJAR
490 IF COLORP= 0 THEN PSET (X1,Y1)
500 IF COLORP= 1 THEN LINE - (X1,Y1),I MOD 2
510 REM REPETIR EL PROCESO
520 GOTO 260
525 GOTO 525
530 NEXT I
540 GOTO 610
550 DATA 0,0,0,100,1,0,10,25,1,300,0,0,0,0,10,25
560 DATA 1,0,40,0,1,0,83,0,1,33,83,0,1,65,35,0
570 DATA 0,0,40,0,1,300,0,0,0,150,4,0,1,30,0,-100
580 DATA 1,0,0,-100,1,0,10,-25,1,300,0,0,0,0,10,-25
590 DATA 1,0,40,0
600 DATA 0,0,0,-100,1,0,0,100,1,30,0,100,1,150,4,0,-1
610 REM CAMBIA LOS DATOS
620 C=C+INC
630 D=D+IND
640 E=E+INE
650 XM=XM*E1+INX
660 YM=YM*E2+INY
670 ZM=ZM+INZ
680 GOTO 200


VARIACIONES PARA M.S.X.

LINEA  80 SCREEN 2
LINEA 130 IE=-15*PI/180
LINEA 620 C=C+IC
LINEA 630 D=D+ID
LINEA 640 E=E+IE
LINEA 650 XM=XM*E1+IX
LINEA 660 YM=YM*E2+IY
LINEA 670 ZM=ZM+IZ
```

Programa 6.1

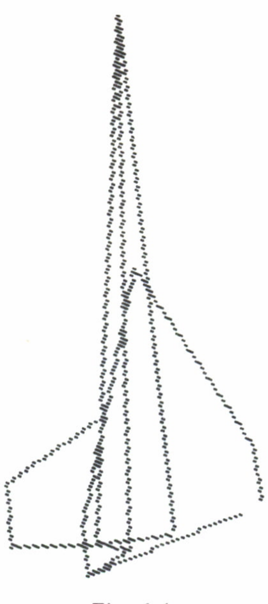


Fig. 6.1

El programa es igual que los del capítulo 5, con la diferencia de que en este caso las transformaciones son variables y cambian cada vez que se repite el bucle, la rotación en el eje Z en el programa anterior.

Otra forma de crear movimiento es no mover el objeto, sino la posición del observador, empleando la proyección generalizada de la que hablamos en el capítulo 3.

Ahora, en vez de cambiar las transformaciones para hacerlas variables, lo que habría que cambiar son los datos relativos a la posición del observador para simular un movimiento de éste.

El siguiente programa hace precisamente eso, va moviendo la posición del observador para simular un movimiento de acercamiento en espiral.

```
10 REM ****************************************
20 REM **                                    **
30 REM **      VERSION PARA IBM              **
40 REM **                                    **
50 REM **      MOVIMIENTO DE OBSERVADOR      **
60 REM **                                    **
70 REM ****************************************
80 SCREEN 1
90 CLS
100 PI=3.141592654#
```

```
110 X0=160:Y0=100
120 DIS=300
130 INA=-5*PI/180:INB=5*PI/180:INDIS=-5
140 A=0*PI/180:B=90*PI/180
150 A1=COS(A):A2=SIN(A)
160 B1=COS(B):B2=SIN(B)
170 FOR I=1 TO 2
180 RESTORE
190 REM VER SI HA ACABADO
200 READ COLORP
210 IF COLORP=-1 THEN GOTO 410
220 REM LEER EL PUNTO
230 READ X,Y,Z
240 X=X*.5:Y=Y*.5:Z=Z*.5
250 REM GIRA
260 REM CALCULAR LA PROYECCION
270 X=A1*X-A2*Z
280 Y=Y*B1-Z*A1*B2-X*A2*B2
290 Z=Z*A1*B1+X*A2*B1+Y*B2
300 PX =DIS*X/(DIS-Z)
310 PY =DIS*Y/(DIS-Z)
320 REM TRASLADAR EL ORIGEN
330 X1=PX+X0
340 Y1=Y0-PY
350 REM VER SI HAY QUE DIBUJAR
360 IF COLORP= 0 THEN PSET (X1,Y1)
370 IF COLORP= 1 THEN LINE - (X1,Y1),I MOD 2
380 REM REPETIR EL PROCESO
390 GOTO 200
400 GOTO 400
410 NEXT I
420 GOTO 490
430 DATA 0,0,0,100,1,0,10,25,1,300,0,0,0,0,10,25
440 DATA 1,0,40,0,1,0,83,0,1,33,83,0,1,65,35,0
450 DATA 0,0,40,0,1,300,0,0,0,150,4,0,1,30,0,-100
460 DATA 1,0,0,-100,1,0,10,-25,1,300,0,0,0,0,10,-25
470 DATA 1,0,40,0
480 DATA 0,0,0,-100,1,0,0,100,1,30,0,100,1,150,4,0,-1
490 REM CAMBIA LOS DATOS
500 A=A+INA
510 B=B+INB
520 DIS=DIS+INDIS
530 GOTO 150
540 YM=YM*E2+INY
550 ZM=ZM+INZ
560 GOTO 140
```

VARIACIONES PARA M.S.X.

LINEA 80 SCREEN 2

```
LINEA 130 IA=-5*PI/180:IB=5*PI/180:IN=-5
LINEA 500 A=A+IA
LINEA 510 B=B+IB
LINEA 520 DIS=DIS+IN
LINEA 540 YM=YM*E2+IY
LINEA 550 ZM=ZM+IZ
```

Programa 6.2

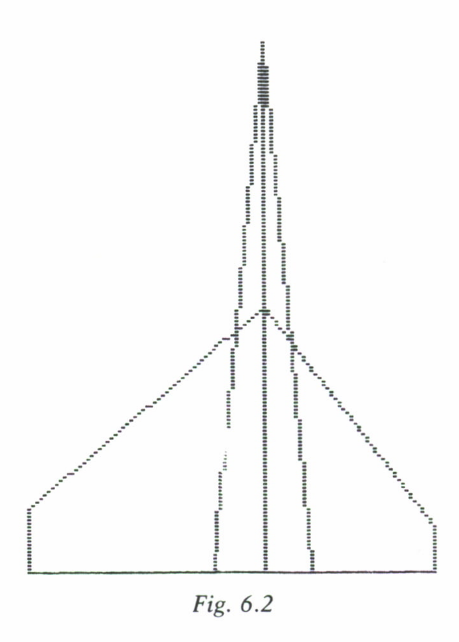


Fig. 6.2

Al ejecutar cualquiera de los dos programas anteriores nos habremos dado cuenta que son muy lentos, esto es, por estar en BASIC, pero los mismos algoritmos podrían ser programados en un lenguaje más rápido, PASCAL, o mejor ensamblador, consiguiendo unos efectos muy realistas.



GENERACION DE CUERPOS DE REVOLUCION

Hay muchos objetos en el mundo real cuyo diseño está basado en la creación de un perfil en dos dimensiones transformándolo en un cuerpo tridimensional girando este perfil sobre un eje determinado.

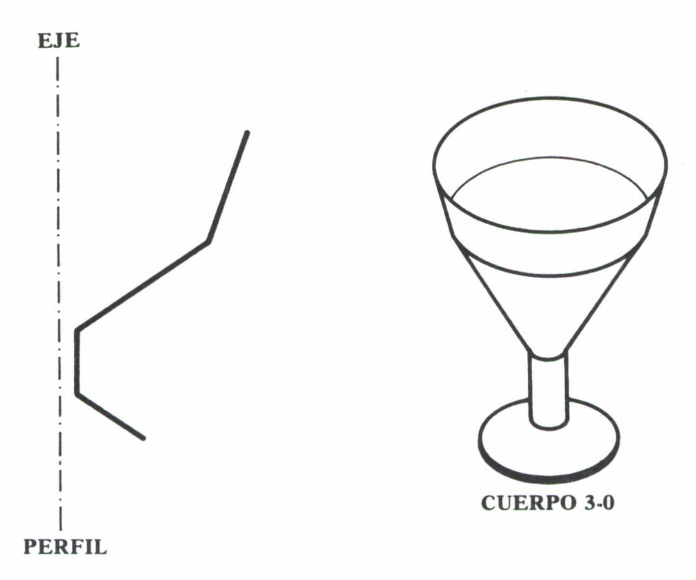


Fig. 6.3

Realizar esto con un ordenador es muy sencillo, solamente hay que preocuparse de llevar un poco de orden a la hora de hacer los cálculos.

Primero hay que almacenar, en una DATA por ejemplo, las coordenadas del perfil, de tal forma que la coordenada Z sea siempre 0 y una vez hecho esto ir rotando este perfil alrededor del eje Z para conseguir las distintas secciones de la figura almacenándolas en una matriz con tres subíndices, tal que el tercer subíndice indica la coordenada X,Y o Z del punto; el segundo indica la sección a la que pertenece el punto y el primero el punto de la sección.

Para realizar los giros sólo hay que aplicar unas sencillas fórmulas a las coordenadas X e Y originales, ya que la coordenada Z permanece constante. Las fórmulas son:

$$X(w) = X\ COS(a)$$
$$Y(w) = Y\ SIN(a)$$

Donde a es un ángulo que varía entre 0 y 360 grados, con un paso de 360/n, siendo n el número de secciones.

Una vez llena la matriz con los puntos del cuerpo que hemos generado, la vamos recorriendo según las filas; la primera coordenada, para pintar los perfiles en la pantalla, y luego la leemos por columnas para unir las distintas secciones entre sí.

```
10 REM ***************************
20 REM **                       **
30 REM ** PROGRAMA AMSTRAD      **
40 REM **                       **
50 REM ** CUERPOS DE REVOLUCION *
60 REM **                       **
70 REM ***************************
80 READ N
90 CLS:CLG
100 xo=320:yo=100
110 LOCATE 1,25
120 INPUT "Numero de secciones ->";m
130 LOCATE 1,25
140 INPUT "distancia focal,sepy";d,oy
150 CLS:CLG
155 LOCATE 10,13:PRINT "MAQUINA EN OPERACION"
160 DIM a(n,m,3)
170 inc=2*PI/m
180 ang=0
190 FOR i=1 TO n
200 FOR j=1 TO 3
210 READ a(i,1,j)
220 NEXT j
230 FOR k=2 TO m
240 ang=ang+inc
250 a1=COS(ang):a2=SIN(ang)
260 x=a(i,1,1):y=a(i,1,2)
270 a(i,k,1)=x*a1
280 a(i,k,2)=y
290 a(i,k,3)=x*a2
300 NEXT k
305 ang=0
310 NEXT i
320 LOCATE 10,13:PRINT SPC(20)
330 FOR j=1 TO m
335 FOR  i=1 TO n
340 x=a(i,j,1)
350 y=a(i,j,2)
360 z=a(i,j,3)
370 GOSUB 530
380 IF i=1 THEN MOVE px,py
390 IF i>1 THEN  DRAW px,py
400 NEXT i
410 NEXT j
430 FOR i=1 TO n
435 FOR j=1 TO m
440 x=a(i,j,1)
450 y=a(i,j,2)
```

```
460 z=a(i,j,3)
470 GOSUB 530
480 IF j=1 THEN MOVE  px,py
490 IF j>1 THEN DRAW px,py
500 NEXT j
501 x=a(i,1,1):y=a(i,1,2):z=a(i,1,3)
502 GOSUB 530:DRAW px,py
510 NEXT i
520 END
530 px=d*x/(d-z)
540 py=d*(y-oy)/(d-z)+oy
550 px=px+xo
560 py=py+yo
570 RETURN
600 DATA 5,80,0,0,10,15,0,10,75,0,100,100,0,130,150,0
```

Programa 6.3

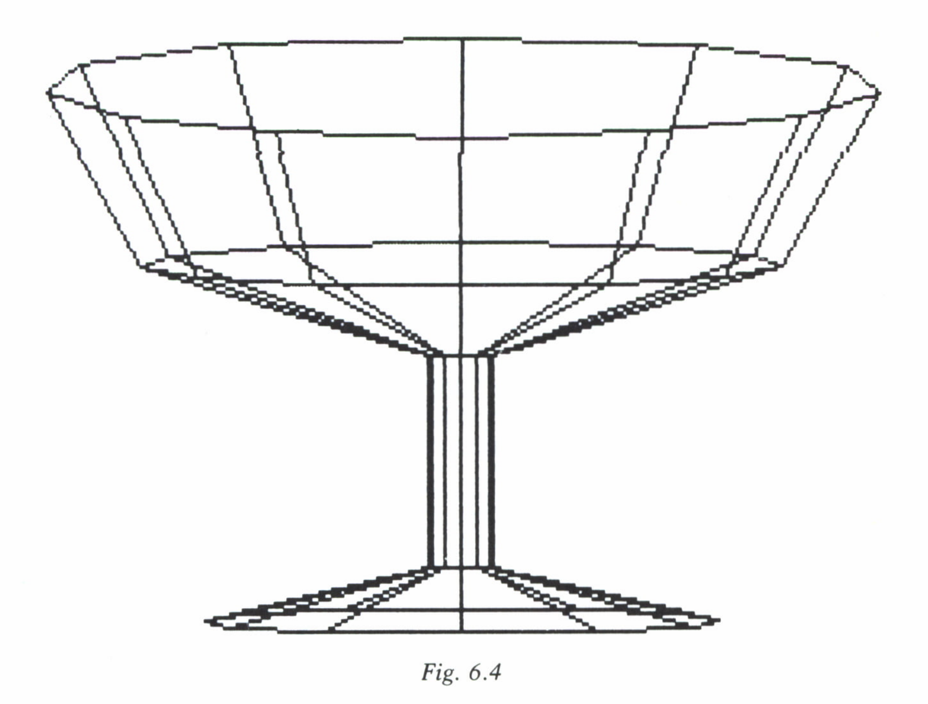


Fig. 6.4

Una modificación interesante en este programa sería que, una vez calculados los puntos del objeto, se almacenaran en una cinta o disco, para más adelante poderlos utilizar sin necesidad de volver a calcular la figura.

Hay que tener en cuenta que este programa realiza una gran cantidad de cálculos y utiliza mucha memoria para almacenar los puntos, por lo que no es conveniente hacer un número de secciones excesivo.

El esquema general del programa sería:

```
LEER NUMERO DE PUNTOS DEL PERFIL
LEER NUMERO DE SECCIONES
LEER LA PERSPECTIVA A UTILIZAR
FOR I=1 a número de puntos del perfil
   LEER COORDENADA DEL PUNTO I
   FOR J=2 a número de secciones
       CALCULA LA PROYECCION Y ALMACENA EN MATRIZ
   FIN DEL BUCLE J
FIN DEL BUCLE I
FOR I=1 a NUMERO DE SECCIONES
                ||    FOR J=1 a NUMERO DE PUNTOS DE UNA SECCION
   UNE          ||        PROYECTA EL PUNTO (I,J)
SECCIONES       ||    FIN DEL BUCLE J
                || FIN DEL BUCLE I

                || FOR I=1 a NUMERO DE PUNTOS DE UNA SECCION
                ||    FOR J=1 a NUMERO DE SECCIONES
  PINTA         ||        PROYECTA EL PUNTO (J,I)
SECCIONES       ||    FIN DEL BUCLE J
                ||    UNE PUNTO FINAL CON PUNTO INICIAL
                || FIN DEL BUCLE I
```



CUERPOS FORMADOS POR TRASLACION

Otra forma de generar cuerpos a partir de figuras en dos dimensiones es por traslación, que consiste en desplazar una figura paralelamente a sí misma una cierta distancia, uniendo vértices homólogos.

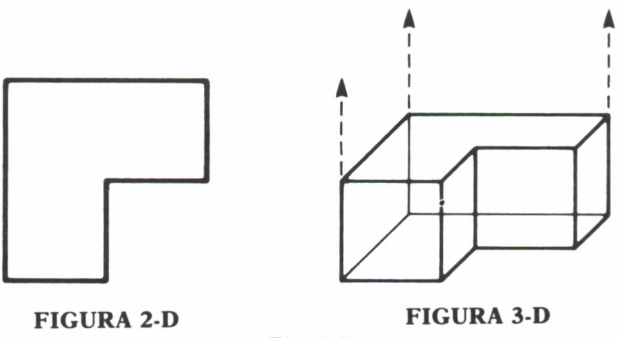


Fig. 6.5

El programa para generar cuerpos de esta manera es muy parecido al programa anterior, pero más sencillo, ya que no hay que hacer giros de ningún tipo; la única operación consiste en copiar las coordenadas del dibujo en dos dimensiones añadiéndole la tercera coordenada que le falta.

La forma de pintarlo sí es completamente igual: primero se pintan las dos superficies y a continuación las unimos mediante trazas verticales.

```
  10 REM ****************************
  20 REM **                        **
  30 REM **PROGRAMA PARA SPECTRUM **
  40 REM **                        **
  50 REM **      TRSLACION         **
  60 REM **                        **
  70 REM ****************************
  80 LET D=100: LET OY=20
  90 INPUT "ALTURA ";A
 100 READ N
 110 DIM P(N,2)
 120 FOR I=1 TO N
 130 READ P(I,1),P(I,2)
 140 NEXT I
 150 FOR J=0 TO 1
 160 LET Y=J*A
 170 FOR I=1 TO N
 180 LET X=P(I,1): LET Z=P(I,2)
 200 GO SUB 1000
 210 IF I=1 THEN  PLOT PX,PY
 220 IF I>1 THEN  DRAW PX-UX,PY-UY
 230 LET UX=PX: LET UY=PY
 240 NEXT I
 250 LET X=P(1,1): LET Z=P(1,2)
 260 GO SUB 1000
 270 DRAW PX-UX,PY-UY
 280 NEXT J
 300 FOR I=1 TO N
 310 LET X=P(I,1): LET Z=P(I,2)
 320 LET Y=0
 330 GO SUB 1000
 340 PLOT PX,PY
 350 LET UX=PX: LET UY=PY
 360 LET Y=A
 370 GO SUB 1000
 380 DRAW PX-UX,PY- UY
 390 NEXT I
 400 STOP
1000 REM RUTINA DE PROYECCION
1010 LET PX=D*X/(D-Z)+100
```

```
1020 LET PY=D*(OY-Y)/(D-Z)+50+OY
1030 RETURN
2000 DATA 7,30,0,15,15,0,40,-15,15,-30,0,-20,20,20,-20
```

Programa 6.4

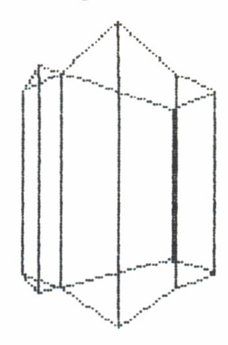


Fig. 6.6

Partiendo de una serie de figuras sobre uno de los planos XY, XZ o YZ, podemos llegar a construir una escena todo lo compleja que queramos.



ESCENAS MULTIPLES

Dibujar una escena en la cual aparezcan una serie de objetos, algunos de ellos repetidos, y en distintas posiciones no es más que repetir tantas veces el proceso de pintar y proyectar una figura como objetos queramos que tenga ésta.

El programa que cierra este capítulo es una muestra de ello.

En él tenemos los datos de dos objetos, un cubo y un avión, y con ellos vamos a realizar una escena compleja.

El procedimiento es el siguiente:

Por cada objeto que queramos dibujar leeremos sus coordenadas originales de las DATA correspondientes y le aplicaremos las transformaciones adecuadas para colocarlo en el lugar del espacio que queramos, procediendo seguidamente a pintarlo.

```
10 REM ****************************************
20 REM **                                    **
30 REM **      VERSION PARA IBM              **
40 REM **                                    **
50 REM **      MUCHO OBJETOS JUNTOS          **
60 REM **                                    **
70 REM ****************************************
```

```
80 SCREEN 1
90 CLS
100 PI=3.141592654#
110 XO=160:YO=100:DIS=1000
120 A=30*PI/180:B=30*PI/180
130 A1=COS(A):A2=SIN(A)
140 B1=COS(B):B2=SIN(B)
150 READ N
160 DIM OBJ(N),DAT(N,3,3)
170 FOR I=1 TO N
180 READ OBJ(I)
190 FOR K=1 TO 3
200 FOR J=1 TO 3
210 READ DAT(I,K,J)
220 NEXT J,K,I
230 FOR I= 1 TO N
240 OB=OBJ(I)/10
250 RESTORE
260 READ A
270 IF -A<>OB THEN GOTO 260
280 C=DAT(I,1,1):D=DAT(I,1,2):E=DAT(I,1,3)
290 SX=DAT(I,2,1):SY=DAT(I,2,2):SZ=DAT(I,2,3)
300 XM=DAT(I,3,1):YM=DAT(I,3,2):ZM=DAT(I,3,3)
310 C=C*PI/180:D=D*PI/180:E=E*PI/180
320 C1=COS(C):C2=SIN(C)
330 D1=COS(D):D2=SIN(D)
340 E1=COS(E):E2=SIN(E)
350 REM VER SI HA ACABADO
360 READ COLORP
370 IF COLORP=-1 THEN GOTO 630
380 REM LEER EL PUNTO
390 READ X,Y,Z
410 REM GIRA
420 Y1=Y*C1+Z*C2:Z=Z*C1-Y*C2:Y=Y1
430 X1=X*D1+Z*D2:Z=Z*D1-X*D2:X=X1
440 X1=X*E1+Y*E2:Y=Y*E1-X*E2:X=X1
450 REM CAMBIA LA ESCALA
460 X=X*SX:Y=Y*SY:Z=Z*SZ
470 REM TRASLADA
480 X=X+XM:Y=Y+YM:Z=Z+ZM
490 REM CALCULAR LA PROYECCION
500 X=A1*X-A2*Z
510 Y=Y*B1-Z*A1*B2-X*A2*B2
520 Z=Z*A1*B1+X*A2*B1+Y*B2
530 PX =DIS*X/(DIS-Z)
540 PY =DIS*Y/(DIS-Z)
550 REM TRASLADAR EL ORIGEN
560 X1=PX+XO
```

```
570 Y1=YO-PY
580 REM VER SI HAY QUE DIBUJAR
590 IF COLORP= 0 THEN PSET (X1,Y1)
600 IF COLORP= 1 THEN LINE - (X1,Y1)
610 REM REPETIR EL PROCESO
620 GOTO 360
630 NEXT I
640 END
650 DATA 9,1,0,0,0,.2,.5,.3,100,0,-100
660 DATA 1,0,0,0,.3,.5,.3,100,0,100
670 DATA 1,0,0,0,.3,.5,.3,100,0,0
680 DATA 1,0,0,0,.2,.5,.3,-100,0,-100
690 DATA 1,0,0,0,.3,.5,.3,-100,0,100
700 DATA 1,0,0,0,.3,.5,.3,-100,0,0
710 DATA 2,0,0,0,1.5,1,1.5,-37,0,-225
720 DATA 3,-45, 90,  0,.25,.25,.25,-50,20,-100
730 DATA 3,15,95,5,.25,.25,.25,50,50,200
740 DATA -.1,0,0,0,0,1,100,0,0,1,100,100,0,1,0,100,0
750 DATA 1,0,100,100,1,0,0,100,1,100,0,100,1,100,100,
    100
760 DATA 1,0,100,100,0,100,100,100,1,100,100,0,0,100,
    0,100
770 DATA 1,100,0,0,0,0,0,100,1,0,0,0,1,0,100,0,-1
780 DATA -.2,0,0,0,0,1,50,0,0,1,50,0,50,1,0,0,50
790 DATA 1,0,0,0,1,25,50,25,1,0,0,50,0,50,0,50
800 DATA 1,25,50,25,1,50,0,0,-1
810 DATA -.3,0,0,0,100,1,0,10,25,1,300,0,0,0,0,10,25
820 DATA 1,0,40,0,1,0,83,0,1,33,83,0,1,65,35,0
830 DATA 0,0,40,0,1,300,0,0,0,150,4,0,1,30,0,-100
840 DATA 1,0,0,-100,1,0,10,-25,1,300,0,0,0,0,10,-25
850 DATA 1,0,40,0
860 DATA 0,0,0,-100,1,0,0,100,1,30,0,100,1,150,4,0,-1

VARIACIONES PARA M.S.X.

LINEA  80 SCREEN 2
```

Programa 6.5

Ya hemos visto algunas aplicaciones de las gráficas en tres dimensiones; a partir de este momento es suficiente con la imaginación del lector para llegar a producir las imágenes más sorprendentes y vistosas.

Evidentemente no hemos sido totalmente exhaustivos ni formales con las técnicas de representación que hemos visto hasta ahora, pero son un

bagaje suficiente como para poder realizar cualquier programa e investigar por nuestra cuenta.

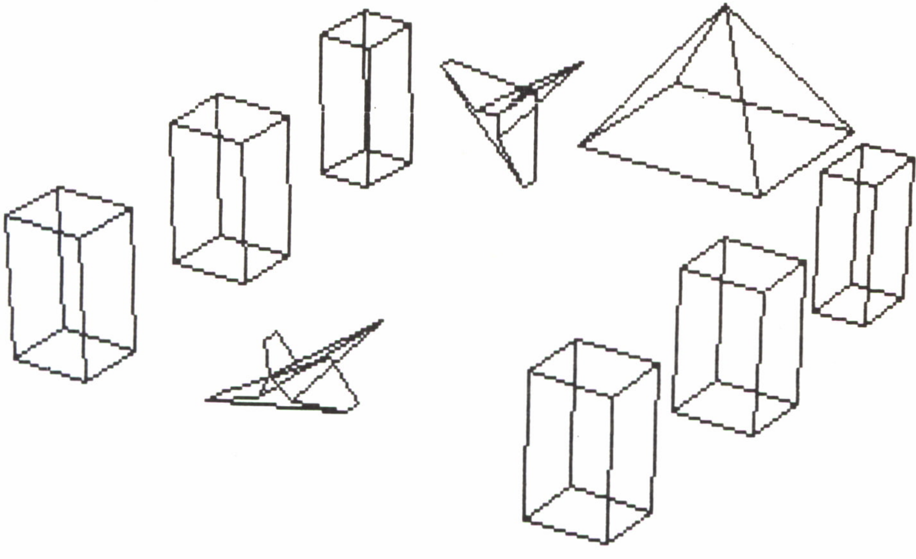
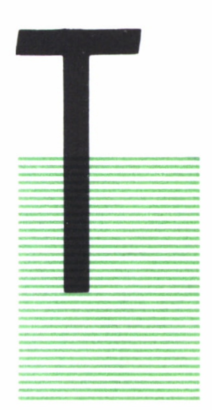


Fig. 6.7

ELIMINACION DE SUPERFICIES OCULTAS 7



TAL y como hemos estado mostrando los objetos en la pantalla, éstos consisten en una especie de «esqueleto» de alambre representando sus aristas y vértices por medio de líneas rectas. Este tipo de representación es la más sencilla que podemos usar, y en nuestro caso casi la única, ya que otras técnicas implican el disponer de terminales gráficos especiales que los ordenadores personales normalmente no tienen.

Esta forma de representación tiene un inconveniente que probablemente el lector ya habrá advertido, y es que hay ocasiones en las que es difícil discernir en profundidad la tercera dimensión, de forma correcta, por dos razones concretas: la primera, por el simple hecho de estar representando una figura de tres dimensiones sobre un plano; y la segunda, porque en ocasiones las caras o superficies que quedarían ocultas en la realidad tapadas por las caras que realmente se ven en la pantalla del ordenador si se dibujan, haciendo que el dibujo se vea confuso.

Por esto, llegamos a uno de los problemas típicos de la representación de objetos en tres dimensiones: la eliminación de las superficies o caras ocultas, esto es, las que quedan tapadas por las caras delanteras que el observador realmente ve.

No hay unos algoritmos o soluciones estándar para resolver este problema, sino que en cada caso se aplica el más conveniente. Ya vimos una forma de hacerlo en el caso particular de diagramas de funciones y de barras, y en el caso de representación de objetos existen una gran cantidad de ellos.

La mayoría de estos algoritmos demandan el uso de ordenadores muy rápidos, dada la gran cantidad de cálculos que tienen que realizar y el hardware especializado que requieren, por lo que nosotros no podremos usarlos en nuestro ordenador personal.

Pero hay un caso especial en el cual sí podremos eliminar las superficies escondidas de nuestros objetos de una forma simple. Cuando los objetos estén formados por caras planas y la forma del cuerpo es «convexa», no tiene picos o vértices hacia adentro.

El funcionamiento de este algoritmo es sencillo. Para cada cara o superficie del cuerpo determinaremos un vector perpendicular a ella y otro vector que una la posición del observador con esta superficie, aplicando a estos dos vectores una fórmula matemática que nos dirá si el ángulo que forman es agudo u obtuso, ya que si es agudo, la cara es visible para el observador, y si es obtuso, no.

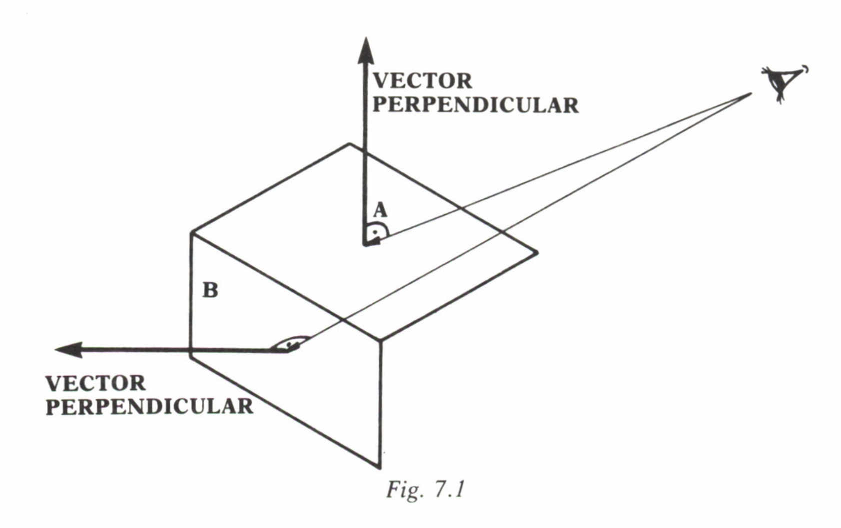


Fig. 7.1

Esta definición nos lleva a considerar que deberemos representar los datos del objeto como algo más que una lista de vértices que debemos unir, ya que ahora tenemos que representar el cuerpo como un conjunto de superficies.

Esto lo haremos representando cada cara del cuerpo como una lista de los vértices que están contenidos en ella, empezando a contar desde cualquier vértice y en el sentido opuesto al de las agujas del reloj vista desde fuera del cuerpo.

El sentido de introducción de los vértices en la lista es importante, ya que si lo hiciéramos al revés estaríamos mirando el objeto desde dentro del cuerpo y no desde fuera.

Estos datos de superficies estarán almacenados en una matriz de dos dimensiones SUP(N,M), donde N es el número de caras que tiene el cuerpo y M el número máximo de vértices que puede llegar a tener una cara. Independientemente tendremos una matriz, VERT(V,3), donde estarán almacenados los vértices del objeto con sus tres coordenadas.

Una vez que tenemos los datos organizados de esta manera, vamos recorriendo la matriz donde están almacenadas las caras y calculamos su vector perpendicular de la siguiente manera:

Calculamos los vectores que unen el primer vértice de la cara con el segundo y tercer vértices, calculando con estos dos vectores su vector perpendicular con las fórmulas:

NX = VI(2) * V2(3) - V2(2) * V1(3)
NY = VI(3) * V2(1) - V2(3) * VI(1)
NZ = VI(1) * V2(2) - V2(1) * VI(2)

Siendo NX,NY,NZ las coordenadas del vector perpendicular y V1(3), V2(3) los vectores calculados anteriormente.

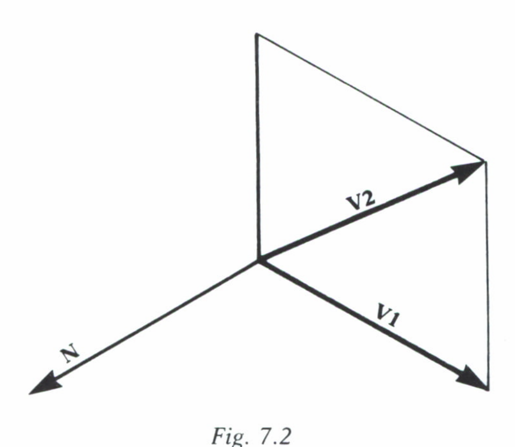


Fig. 7.2

Una vez calculado el vector normal, calculamos el vector que une la posición del observador con el primer vértice de la superficie, LX,LY,LZ, y realizamos el siguiente cálculo:

VE = NX * LX + NY * LY + NZ * LZ

Si el valor de VE es positivo o 0, entonces la superficie se ve y la pintamos de la manera ya habitual, y si VE es negativo, entonces es que la superficie no se ve y no hay que dibujarla.

Para calcular el vector normal hemos utilizado una operación vectorial llamada producto vectorial, cuyo resultado es otro vector perpendicular al plano formado por los dos vectores iniciales.

La variable VE la calculamos también con una operación llamada producto escalar, cuyo resultado es un número; positivo, si el ángulo que forman los dos vectores es agudo, y negativo, si el ángulo es obtuso.

Los vectores V1, V2 y L los calculamos restando las coordenadas de los dos puntos que unen los extremos del vector, resultando así las coordenadas de cada uno de los vectores.

Todo lo que hemos explicado hasta ahora, podemos verlo de forma práctica en el siguiente programa:

```
10 REM ********************************
20 REM **                            **
30 REM **   PROGRAMA PARA AMSTRAD    **
40 REM **                            **
50 REM **      VISTAS Y OCULTAS      **
60 REM **                            **
70 REM ********************************
80 MODE 1
90 CLS:CLG
100 REM INCIALIZA LOS VALORES
110 XO=270:YO=200
120 DIM VEC1(3),VEC2(3)
130 REM LEER EL NUM. SUPERFICIES
140 READ N
150 REM LEER EL NUM. VERTICES EN FIGURA
160 READ M
170 REM LEER EL NUM. DE VERTICE POR SUP.
180 READ V
190 DIM SUP(N,V)
200 FOR I=1 TO N
210 FOR J=1 TO V
220 READ SUP(I,J)
230 NEXT J
240 NEXT I
250 DIM VERT(M,3)
260 FOR I=1 TO M
270 FOR J=1 TO 3
280 READ VERT(I,J)
290 NEXT J
300 NEXT I
310 D=300:OX=300:OY=300
320 REM RECORRE TODAS LAS SUPERFICIES
330 FOR I=1 TO N
340 REM CALCULA DOS VECTORES INCLUIDOS EN SUP.
350 FOR  J=1 TO 3
360 VEC1(J)=VERT(SUP(I,2),J)-VERT(SUP(I,1),J)
370 VEC2(J)=VERT(SUP(I,3),J)-VERT(SUP(I,1),J)
380 NEXT J
390 REM CALCULA EL VECTOR NORMAL A LA SUP.
400 NX=VEC1(2)*VEC2(3)-VEC2(2)*VEC1(3)
410 NY=VEC1(3)*VEC2(1)-VEC2(3)*VEC1(1)
```

```
420 NZ=VEC1(1)*VEC2(2)-VEC2(1)*VEC1(2)
430 REM CALCULA LINEA ENTRE PUNTO DE VISTA Y SUP.
440 LX=OX-VERT(SUP(I,1),1)
450 LY=OY-VERT(SUP(I,1),2)
460 LZ=D-VERT(SUP(I,1),3)
470 REM OBSERVA SI ES VISIBLE
480 VE=NX*LX+NY*LY+NZ*LZ
490 REM SI NO SE VE NO PINTAR
500 IF VE<0 THEN GOTO 590
510 FOR  J=1 TO V
520 GOSUB 670
530 IF  J=1 THEN MOVE PX,PY
540 IF  J>1 THEN DRAW PX,PY
550 NEXT J
560 J=1
570 GOSUB 670
580 DRAW PX,PY
590 NEXT I
600 END
610 DATA 6,8,4
620 DATA 1,2,3,4,2,6,7,3,3,7,8,4
630 DATA 6,5,8,7,1,5,6,2,1,4,8,5
640 DATA 100,0,0,100,100,0,100,100,100
650 DATA 100,0,100,0,0,0,0,100,0
660 DATA 0,100,100,0,0,100
670 REM RUTINA DE PROYECTAR UN VERTICE
680 X=VERT(SUP(I,J),1)
690 Y=VERT(SUP(I,J),2)
700 Z=VERT(SUP(I,J),3)
710 PX=D*(X-OX)/(D-Z)+OX+XO
720 PY=D*(Y-OY)/(D-Z)+OY+YO
730 RETURN
```

Programa 7.1

Este programa en principio dibuja un cubo eliminando las superficies que no se ven, pero podemos dibujar cualquier figura mientras se cumpla la condición de que ésta sea convexa.

Para preparar las matrices de un objeto realizamos el siguiente proceso:

1.° Tabla de vértices, que corresponde con la matriz VERT.

2.° Tabla de superficies, en las que la fila indica la superficie del cuerpo y la columna el vértice de esa superficie. Esta tabla se corresponde con la matriz SUP.

EJEMPLO DE TABLAS PARA EL CUBO

1. Vértices

Vert. N.°	X	Y	Z
1	100	0	0
2	100	100	0
3	100	100	100
4	100	0	100
5	0	0	0
6	0	100	0
7	0	100	100
8	0	0	100

2. Superficies

Sup. N.°	VERTICES			
1	1	2	3	4
2	2	6	7	3
3	3	7	8	4
4	6	5	8	7
5	1	5	6	2
6	1	4	8	5

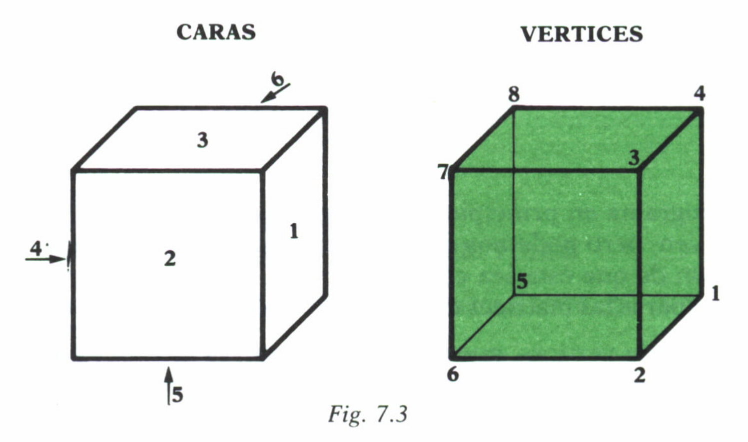


Fig. 7.3

Estas dos tablas nos dan las matrices VERT y SUP del cubo.

Con todas las técnicas y programas que hemos visto en este libro podemos representar cualquier cuerpo o escena del espacio tridimensional sobre la pantalla de nuestro ordenador de una forma sencilla.

Hay otro tipo de técnicas mucho más avanzadas, que permiten el modelado de las superficies de los objetos y tener en cuenta incluso su textura, pero no es posible tratarlas aquí, ya que los ordenadores personales no disponen de capacidades gráficas muy avanzadas.

De todas formas, las técnicas aquí descritas son el principio del diseño de objetos tridimensionales en el ordenador, utilizándose unas técnicas parecidas en CAD.

Como proyecto final de aplicación de todo lo que hemos aprendido sería una buena idea diseñar un programa de diseño interactivo de figuras tridimensionales con el ordenador, de forma que una vez diseñadas pudiéramos almacenar los datos de la figura en una cinta o un disco para poder usarlos más adelante en cualquier programa.

Este programa nos permitiría, mediante una serie de teclas, el trazar líneas y unirlas entre sí para diseñar el objeto, viendo el resultado de cada operación en la pantalla, con distintos puntos de vista y perspectivas.

APENDICE 1

OPERACIONES CON MATRICES



Repasaremos las operaciones más usuales que se realizan con matrices.



SUMA Y RESTA

Para sumar y restar dos matrices, ambas deben tener las mismas dimensiones, siendo la matriz resultado de igual dimensión.

La forma de realizar estas operaciones es muy sencilla. Cada elemento de la matriz resultado será la suma, o resta, de los elementos que ocupen su misma posición de las matrices operando.

Veamos un ejemplo para aclararlo:

$$\left\|\begin{matrix}1 & 2\\ 3 & 4\end{matrix}\right\| + \left\|\begin{matrix}5 & 6\\ 7 & 8\end{matrix}\right\| = \left\|\begin{matrix}1+5 & 2+6\\ 3+7 & 4+8\end{matrix}\right\| = \left\|\begin{matrix}6 & 8\\ 10 & 12\end{matrix}\right\|$$

Este procedimiento también es válido para vectores, que son matrices de dimensiones 1 × 2.

$$\|\ 5\ \ 6\ \ 7\ \ 8\ \| - \|\ 1\ \ 2\ \ 3\ \ 4\ \| = \|\ 5\text{-}1\ \ 6\text{-}2\ \ 7\text{-}3\ \ 8\text{-}4\ \| = \|\ 4\ \ 4\ \ 4\ \ 4\ \|$$



MULTIPLICACION POR UNA CONSTANTE

Para multiplicar una constante por una matriz se multiplica cada elemento de la matriz por dicha constante.

Veamos un ejemplo:

$$3 * \left\|\begin{matrix}1 & 2\\ 3 & 4\end{matrix}\right\| = \left\|\begin{matrix}1*3 & 2*3\\ 3*3 & 4*3\end{matrix}\right\| = \left\|\begin{matrix}3 & 6\\ 9 & 12\end{matrix}\right\|$$



MULTIPLICACION DE DOS MATRICES

En esta operación se necesita que el número de columnas de la primera matriz sea igual al de filas de la segunda matriz.

La matriz resultante tendrá tantas filas como tuviese la primera y tantas columnas como la segunda.

La forma de realizar esta operación es que el elemento situado en la fila i y en la columna j de la matriz resultado será igual a la suma de multiplicar los elementos de la columna j de la segunda matriz.

Observemos unos ejemplos:

$$\begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix} * \begin{Vmatrix} 5 & 6 \\ 7 & 8 \end{Vmatrix} = \begin{Vmatrix} 1*5+2*7 & 1*6+2*8 \\ 3*5+4*8 & 3*6+4*8 \end{Vmatrix} = \begin{Vmatrix} 9 & 22 \\ 47 & 50 \end{Vmatrix}$$

$$\begin{Vmatrix} 1 & 2 \end{Vmatrix} * \begin{Vmatrix} 3 & 4 \\ 5 & 6 \end{Vmatrix} = \begin{Vmatrix} 1*3+2*5 & 1*4+2*6 \end{Vmatrix} = \begin{Vmatrix} 13 & 16 \end{Vmatrix}$$

$$\begin{Vmatrix} 1 & 2 \end{Vmatrix} * \begin{Vmatrix} 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{Vmatrix}$$

No puede realizarla por no tener igual número de filas que columnas.

Hay que destacar que no es una operación conmutativa, es decir, sí importa el orden de los factores:

$$A*B \neq B*A$$

Observemos dos ejemplos:

$$\begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix} * \begin{Vmatrix} 5 & 6 \\ 7 & 8 \end{Vmatrix} = \begin{Vmatrix} 1*5+2*7 & 1*6+2*8 \\ 3*5+4*7 & 3*6+4*8 \end{Vmatrix} = \begin{Vmatrix} 19 & 22 \\ 47 & 50 \end{Vmatrix}$$

$$\begin{Vmatrix} 5 & 6 \\ 7 & 8 \end{Vmatrix} * \begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix} = \begin{Vmatrix} 5*1+6*3 & 5*2+6*4 \\ 7*1+8*3 & 7*2+8*4 \end{Vmatrix} = \begin{Vmatrix} 23 & 34 \\ 31 & 46 \end{Vmatrix}$$

Como se observa, los resultados son totalmente diferentes.

$$\begin{Vmatrix} 1 & 2 \end{Vmatrix} * \begin{Vmatrix} 3 & 4 \\ 5 & 6 \end{Vmatrix} = \begin{Vmatrix} 1*3+5*2 & 1*4+2*6 \end{Vmatrix} = \begin{Vmatrix} 13 & 16 \end{Vmatrix}$$

$$\begin{Vmatrix} 3 & 4 \\ 5 & 6 \end{Vmatrix} * \begin{Vmatrix} 1 \\ 2 \end{Vmatrix} = \begin{Vmatrix} 3*1 + 4*2 \\ 5*1 + 6*2 \end{Vmatrix} = \begin{Vmatrix} 11 \\ 17 \end{Vmatrix}$$

En este caso las matrices resultado ni siquiera tienen igual dimensión.



DETERMINANTES

Existe un cálculo que asigna un número a una matriz cuadrada, es decir, que tiene igual número de filas que de columnas.

Veamos los casos 2 × 2 y 3 × 3:

En el caso de dos por dos hay un truco:

$$\left\| \begin{matrix} a & b \\ c & d \end{matrix} \right\| = a*d - b*c$$

Veamos un ejemplo:

$$\left\| \begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \right\| = 1*4 - 2*3 = 4*6 = -2$$

En el caso de tres por tres la regla es la siguiente:

$$\left\| \begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix} \right\| = a*e*i + b*f*g + d*h*c - c*e*g - h*f*a - b*d*i$$

Veamos un ejemplo:

$$\left\| \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \right\| = 1*5*9 + 2*6*7 + 4*8*3 - 3*5*7 - 6*8*1 - 4*2*9 = 45 + 84 - 92 - 105 - 48 - 72 = -4$$





Incluimos una pequeña tabla de equivalencias de comandos gráficos para permitir una mayor facilidad en la adaptación de los programas a los distintos programas.

Hay que tener en cuenta las diferentes resoluciones de las pantallas de los ordenadores; por ello, se deberán multiplicar los datos por constantes, tal como se realiza en alguno de los programas.

La constante para pasar del Amstrad al IBM es de 0,5; del Amstrad al Spectrum será de 0,3; del IBM al Spectrum será de 0,7. En los demás casos no es muy importante, pues la imagen será más pequeña, pero no se saldrá de la pantalla.

Trazar una recta que una los puntos X1,Y1 y X2,Y2:

IBM: LINE (X1,Y1) - (X2,Y2)
Amstrad: MOVE X1,Y1: DRAW X2,Y2
Spectrum: PLOT X1,Y1: DRAW X2-X1,Y2-Y1

Trazar una recta que una el punto X,Y con el último dibujo x1,y1:

Referido al origen de coordenadas:

IBM: LINE - (X,Y)
Amstrad: DRAW X,Y
Spectrum: DRAW X-X1, Y-Y1

Referido al último punto dibujado:

IBM: LINE STEP - (X,Y)
Amstrad: DRAWR X,Y
Spectrum: DRAW X,Y

Borrar una recta que una X1,Y1 con X2,Y2:

IBM: LINE (X1,Y1) - (X2,Y2), 0
Amstrad: MOVE X1,Y1: PEN 0: DRAW X2,Y2: PEN 1
Spectrum: INK 0: PLOT X1,Y1: DRAW X2,Y2: INK 1

Dibujar un solo punto X,Y:

IBM: PSET (X,Y)
Amstrad: PLOT X,Y
Spectrum: PLOT X,Y

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

INDICE GENERAL

5 AÑADA PERIFERICOS A SU ORDENADOR

Breve descripción de varios periféricos que facilitan la comunicación con el ordenador personal, con algunos ejemplos de fácil construcción: ratón, lápiz óptico, marco para pantalla táctil...

6 GRAFICOS ANIMADOS CON EL ORDENADOR

En este libro las técnicas utilizadas para la animación son el resultado de unas pocas ideas básicas muy sencillas de comprender. Descubrirá los trucos y secretos de movimientos, choques, rebotes, explosiones, disparos, saltos, etc.

7 JUEGOS INTELIGENTES EN MICROORDENADORES

Los ordenadores pueden enfrentarse de forma «inteligente» ante puzzles y otros tipos de juegos. Esto es posible gracias al nuevo enfoque que ha dado la IA a la tradicional teoría de juegos.

8 PERIFERICOS INTERACTIVOS PARA SU ORDENADOR

Descripción detallada de la forma de construir, paso a paso y en su propia casa, dispositivos electrónicos que aumentarán la potencia y facilidad de uso de su ordenador: tableta digitalizadora, convertidores de señales analógicas, comunicaciones entre ordenadores.

9 DIBUJOS TRIDIMENSIONALES EN EL ORDENADOR PERSONAL

Compruebe que también con su ordenador personal puede llegar a diseñar y calcular imágenes en tres dimensiones con técnicas semejantes a las utilizadas por los profesionales del dibujo con equipos mucho más sofisticados.

10 PRACTIQUE MATEMATICAS Y ESTADISTICA CON EL ORDENADOR

En este libro se repasan los principales conceptos de las Matemáticas y la Estadística, desde un punto de vista eminentemente práctico y para su aplicación al ordenador personal. Se basan los diferentes textos en la presentación de pequeños programas (que usted podrá introducir en su ordenador personal).

11 CRIPTOGRAFIA: LA OCULTACION DE MENSAJES Y EL ORDENADOR

En este libro se presentan las técnicas de ocultacion de mensajes a través de la criptografía desde los primeros tiempos hasta la actualidad, en que el uso de los computadores ha proporcionado la herramienta necesaria para llegar al desarrollo de esta ciencia.

12 APL: LENGUAJE PARA PROGRAMADORES DIFERENTES

APL es un lenguaje muy potente que proporciona gran simplicidad en el desarrollo de programas y al mismo tiempo permite programar sin necesidad de conocer todos los elementos del lenguaje. Por ello es ideal para quienes reúnan imaginación y escasa formación en Informática.

13 PRACTIQUE CIENCIAS NATURALES CON EL ORDENADOR

Ejemplos sencillos para practicar con el ordenador. Casos curiosos de la Naturaleza en forma de programas para su ordenador personal.

14 COMO SIMULAR CIRCUITOS ELECTRONICOS EN EL ORDENADOR

Introducción a los diferentes métodos que se pueden emplear para simular y analizar circuitos electrónicos, mediante la utilización de diferentes lenguajes.

15 LOS LENGUAJES DE LA INTELIGENCIA ARTIFICIAL

Libro en que se describen los lenguajes específicos para la «elaboración del saber» y los entornos de programación correspondientes. El conocimiento de estos lenguajes, además de interesante en sí mismo, es sumamente útil para entender todo lo que la Informática Artificial supondrá para el futuro de la Informática.

16 PRACTIQUE FISICA Y QUIMICA CON SU ORDENADOR

Libro eminentemente práctico para realizar pequeños «experimentos» con su ordenador y distraerse de un modo útil.

17 EL ORDENADOR Y LA LITERATURA

En este libro se examinan procesadores de textos, programas de análisis literario y una curiosa aplicación desarrollada por el autor: APOLO, un programa que compone estructuras poéticas.

18 COMO ELEGIR UNA HOJA ELECTRONICA DE CALCULO

En este título se estudian las diferentes versiones existentes de esta aplicación típica, desde el punto de vista de su utilidad para, en función de las necesidades de cada usuario y del ordenador de que dispone, poder elegir aquella que más se adecúe a cada caso.

19 ECONOMIA DOMESTICA CON EL ORDENADOR PERSONAL

Breve introducción a la contabilidad de doble partida y su aplicación al hogar, con explicaciones de cómo utilizar el ordenador personal para facilitar los cálculos, mediante un programa especialmente diseñado para ello.

20 ¿MAQUINAS MAS EXPERTAS QUE LOS HOMBRES?

Después de situar los «sistemas expertos» en el contexto de la inteligencia artificial y describir su construcción, su funcionamiento, su utilidad, etc., se analiza el papel que pueden tener en el futuro (y presente, ya) de la Informática.

21 PRACTIQUE HISTORIA Y GEOGRAFIA CON SU ORDENADOR

Libro interesante para los aficionados a estas ciencias, a quienes presenta una nueva visión de cómo utilizar el microordenador en su estudio.

22 ERGONOMIA: COMUNICACION EFICIENTE HOMBRE-MAQUINA

Análisis de la comunicación entre el hombre y la máquina, y estudio de diferentes soluciones que tienden a facilitarla lo más posible.

23 EL ORDENADOR Y LA ASTRONOMIA

Los cálculos astronómicos y el conocimiento del firmamento en un libro apasionante y curioso.

24 VISION ARTIFICIAL. TRATAMIENTO DE IMAGENES POR ORDENADOR

El procesado de imágenes es un campo de reciente y rápido desarrollo con importantes aplicaciones en áreas tan diversas como la mejora de imágenes biomédicas, robóticas, teledetección y otras aplicaciones industriales y militares. Se presentan los principios básicos, los sistemas y las técnicas de procesado más usuales.

25 LA ESTACION TERMINAL PERSONAL

Las modernas técnicas de comunicación van permitiendo que las grandes capacidades de proceso y el acceso a bases de datos de gran tamaño estén cada día más al alcance de cada usuario (fuera ya de los Centros de Proceso de Datos).

26 EL ORDENADOR COMO MAQUINA DE ESCRIBIR INTELIGENTE

Descripción de los sistemas de tratamiento de textos existentes, análisis comparativos y estudio de posibilidades de cada uno de ellos. Guía práctica para la elección del presente paquete que más se adecúe a nuestras necesidades y al ordenador personal de que dispongamos.

27 EL LENGUAJE C, PROXIMO A LA MAQUINA

Lenguaje de programación que se está imponiendo en los microordenadores más grandes, tanto por su facilidad de aprendizaje y uso, como por su enorme potencia y su adecuación a la programación estructurada. Vinculado íntimamente al sistema operativo UNIX es uno de los lenguajes de más futuro entre los que utilizan los micros personales.

28 EL ORDENADOR COMO INSTRUMENTO MUSICAL Y DE COMPOSICION

Análisis de cómo se puede utilizar el ordenador para la composición o interpretación de música. Libro eminentemente práctico, con numerosos ejemplos (que usted podrá practicar en su ordenador casero) y lleno de sugerencias para disfrutar haciendo de su ordenador un verdadero instrumento musical.

29 LA CREATIVIDAD EN EL ORDENADOR. EXPERIENCIAS EN LOGO

El LOGO es un lenguaje enormemente capacitado para la creación principalmente gráfica y en especial para los niños. En este sentido se han desarrollado numerosas experiencias. En el libro se analizan estas experiencias y las posibilidades del LOGO en este sentido, así como su aplicación a su ordenador casero para que usted mismo (o con sus hijos) pueda repetirlas.

30 SISTEMAS OPERATIVOS: EL SISTEMA NERVIOSO DEL ORDENADOR

Características de diversos sistemas operativos utilizados en los ordenadores personales y caseros. Se trata de llegar al conocimiento, ameno, aunque riguroso, de la misión del sistema operativo de su ordenador, para que usted consiga sacar mayor rendimiento a su equipo.

NOTA: Ediciones Siglo Cultural, S. A., se reserva el derecho de modificar, sin previo aviso, el orden, título o contenido de cualquier volumen de la colección.

Uno de los avances más espectaculares de la informática en los últimos años ha sido el diseño y animación de objetos tridimensionales mediante ordenadores.

Gracias a unas técnicas semejantes a las utilizadas en equipos sofisticados de diseño gráfico pueden llegar a conseguirse imágenes tridimensionales en el ordenador. Basta con aplicar una serie de conceptos básicos de la geometría para convertirlo en una herramienta de sorprendentes posibilidades.